Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой

_____ О.В. Непомнящий
подпись инициалы, фамилия
«____» _____ 2018 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Лабораторный онлайн стенд для цифровой обработки сигнала
Тема

09.04.01 Информатика и вычислительная техника
код и наименование направления

09.04.01.01 Высокопроизводительные вычислительные системы
код и наименование магистерской программы

Научный руководитель	_____	доцент, канд. техн. наук	М.С. Медведев
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	_____		А.А. Бакшеев
	подпись, дата		инициалы, фамилия
Рецензент	_____		
	подпись, дата	должность, ученая степень	инициалы, фамилия
Нормоконтролер	_____		В.И. Иванов
	подпись, дата		инициалы, фамилия

Красноярск 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Онлайн-стенд создается для образовательного процесса, подходит для выполнения лабораторных работ, исследований по дисциплине Цифровая обработка сигнала.

Цифровая обработка сигналов (ЦОС) – это одно из наиболее динамично развиваемых и перспективных направлений современной радиотехники. Важнейшими свойствами ЦОС являются высокая точность, технологичность, нечувствительность к дестабилизирующим факторам, функциональная гибкость. Поэтому удельный вес ЦОС в радиоэлектронных устройствах и системах по мере повышения ее быстродействия и снижения стоимости все более возрастает [1].

Интернет-технологии, что прочно вошли в нашу жизнь и стали для человека чем-то естественным, с каждым годом развиваются все более стремительно. Они предлагают удобство во всех сферах, как в повседневной жизни, так и в профессиональной сфере. Сегодня веб-приложения уже не уступают по популярности своим предшественникам – настольным компьютерным приложениям. Как показывает анализ современного состояния исследований и разработок, использование Интернет-технологий при создании для рассматриваемых задач имеет ряд преимуществ по сравнению с традиционными настольными системами – доступность предлагаемых решений большому числу пользователей, упрощение процесса установки и распространения программного обеспечения, снижение его стоимости, возможность интеграции со сторонними приложениями.

Студенты часто сталкиваются с проблемой выполнения лабораторных работ по дисциплине ЦОС, поскольку имеющиеся программные средства для выполнения данного вида работ дорогостоящие и не являются кроссплатформенными. Следовательно, не каждый может ими воспользоваться. Одной из таких программ является Matlab.

Лабораторный онлайн-стенд предназначен как для локального, так и для дистанционного выполнения лабораторных работ и исследований. Предоставляет методы генерации сигналов, с возможностью проведения частотного анализа для оценки влияния на спектр различных параметров, а также использование фильтров с целью выделения определенных частот этого сигнала. Главными преимуществами сервиса является кроссплатформенность и доступность в любое время.

ЦЕЛЬ

Проект направлен на решение проблемы качественного и наглядного отображения данных связанных с дисциплиной ЦОС, а также для создания удобного инструмента для выполнения лабораторных работ, исследований по данной дисциплине. Онлайн-стенд предоставляет методы генерации сигналов, с возможностью проведения частотного анализа для оценки влияния на спектр различных параметров.

ЗАДАЧИ

Разработать web-приложение и приложение для мобильных устройств позволяющее решать следующие задачи:

- генерировать сигналы различных форм по заданным параметрам;
- проводить спектральный анализ сигнала;
- выполнять исследования с применением оконных функций;
- формировать произвольный сигнал;
- дискретная фильтрация сигнала.

1 Анализ архитектуры системы

1.1 Анализ аналогов приложения

1.1.1 Learn Digital Signal Processing Full

Цифровая обработка сигналов является важной отраслью электроники и телекоммуникационных технологий, которая занимается импровизацией надежности и точности цифровой связи с использованием множества методов. В приложении Learn Digital Signal Processing разъясняются основные понятия цифровой обработки сигналов в простой и легкой для понимания форме. Learn Digital Signal Processing - предназначен для студентов электроники и вычислительной техники. Кроме того, данное приложение полезно для любого читателя, который хотел бы понять больше о различных сигналах, системах и методах для обработки цифрового сигнала.

В приложении Digital Signal Processing Full, показана конструкция фильтра с использованием концепции DSP. Learn Digital Signal Processing имеет хороший баланс между теорией и математической строгостью. Перед тем, как приступить к использованию Learn Digital Signal Processing Full, пользователи должны иметь общее представление о дискретных математических структурах.

Рисунок 1 – Приложение Learn Digital Signal Processing

Данное приложение позволяет ознакомиться с основными понятиями цифровой обработки сигнала. Основным минусом является, отсутствие возможности практических действий в исследовании цифровой обработки сигнала.

1.1.2 DSP CALC

Приложение позволяет решать следующие задачи:

- линейная свертка;
- круговая свертка;

- автокорреляция;
- взаимная корреляция;

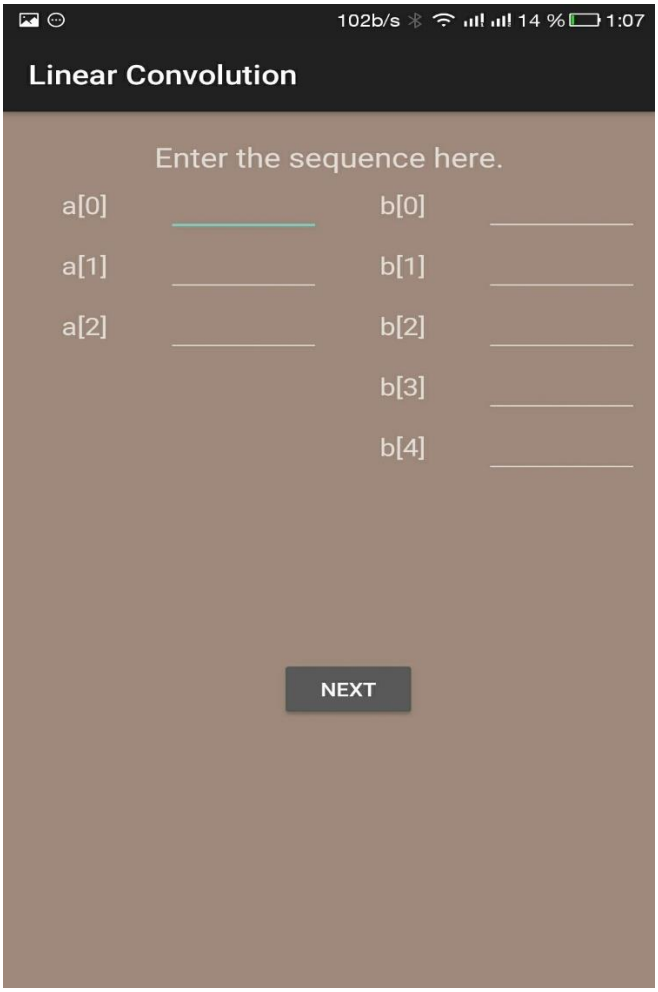

Рисунок 2 – Приложение DSP CALC

Были замечены следующие недостатки:

- отсутствие возможности наглядного исследования сигнала;
- отсутствие удобного интерфейса;
- перевода для русскоязычной аудитории пользователей.

1.1.3 DSP Calculator+

Цифровая обработка сигналов представляет собой способы обработки сигналов на основе численных методов. Некоторые из основных операций представлены в данном приложении:

- линейная свертка;
- круговая свертка;
- автоматическая корреляция;
- взаимной корреляции;
- дискретное преобразование Фурье (ДПФ);
- обратное дискретное преобразование Фурье (ОДПФ).

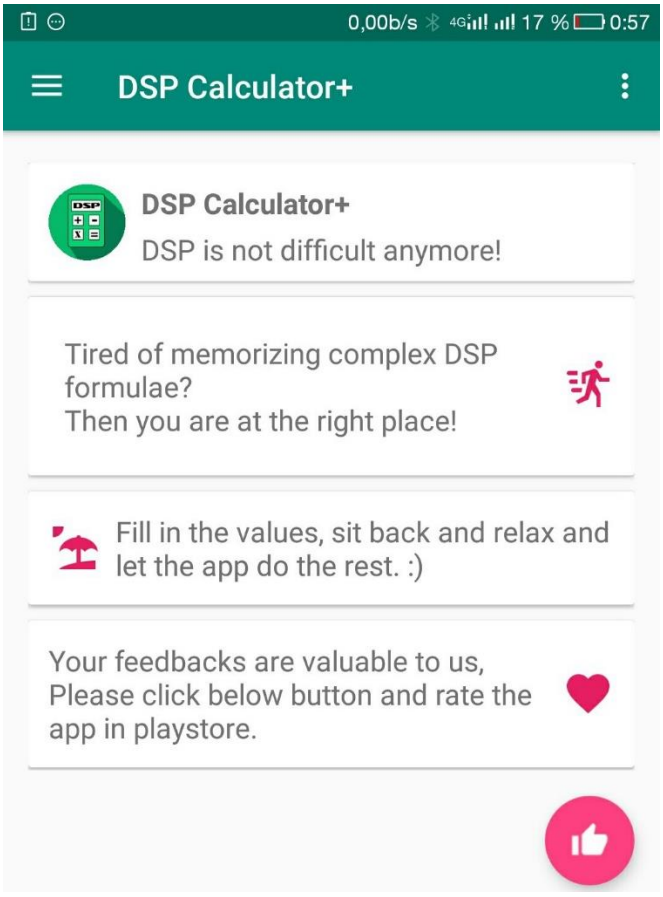

Рисунок 3 – Приложение DSP Calculator+

Данное приложение позволяет провести некоторые основные операции цифровой обработки сигнала. Основным недостатком данного приложения является отсутствие наглядного отображения данных, интуитивно понятного интерфейса, перевода для русской аудитории пользователей.

1.2 Анализ предметной области

Для решения поставленных задач необходимо рассмотреть основные понятия дисциплины цифровая обработка сигнала.

1.2.1 Цифровая обработка сигнала

По определению цифровая обработка сигнала – это обработка цифровых сигналов цифровыми методами и цифровыми средствами [1].

Под цифровым сигналом понимается любая пронумерованная последовательность чисел (цифровых кодов), например, 3, 7, 11, 9, …, в том числе значений оцифрованного аналогового сигнала, являющаяся функцией дискретного аргумента (например, порядкового номера, расстояния или по умолчанию – времени) [1].

Методами ЦОС являются математические соотношения или алгоритмы, в соответствии с которыми выполняются вычислительные операции над цифровыми сигналами. К ним относятся алгоритмы цифровой фильтрации, спектрально-корреляционного анализа, модуляции и демодуляции сигналов, адаптивной обработки и др. Алгоритмы ЦОС, в отличие от других вычислений на ЭВМ, предусматривают, как правило, их выполнение в реальном масштабе времени [1].

1.2.2 Спектральный анализ

Спектральный анализ заключается в определении частотного состава сигнала и оценивании его спектральных характеристик, являющихся функциями частоты [1].

С помощью спектрального анализа решаются задачи обнаружения, разрешения и оценки параметров сигналов, сжатия данных, идентификации объектов, распознавания образов – речи, изображений и т.д. Спектральный анализ случайных сигналов направлен на выявление скрытых периодичностей и статистических (корреляционных) связей. Анализ амплитудных и фазовых спектров периодических (регулярных) сигналов и сигналов конечной длительности называют также гармоническим анализом [1].

Различают следующие методы спектрального анализа: фильтровые (полосовой анализ), бесфильтровые (на основе дискретных преобразований Фурье, Уолша, Хаара, Хартли, вейвлет-анализ), параметрические (на основе параметрических моделей случайных сигналов), методы текущего, скользящего и скачущего анализа, последовательного и параллельного, одноканального и многоканального, выполняемого в реальном времени [1].

Особенности классического спектрального анализа на основе дискретного преобразования Фурье (ДПФ) связаны с оценкой спектра сигнала по его реализациям конечной длины, т. е. на конечном интервале наблюдения. При этом полагается, что за пределами этого интервала сигнал является периодическим продолжением считанной реализации с периодом, равным или большим ее длительности. Широкое применение анализаторов спектра на основе ДПФ обусловлено наличием высокоэффективных вычислительных алгоритмов быстрого преобразования Фурье (БПФ) [1].

1.2.3 Преобразование Фурье

Преобразование Фурье является математической основой спектрального анализа сигнала.

Алгоритмы быстрого преобразования Фурье (БПФ) − это способы быстрого вычисления ДПФ $X(jk) = \sum_{n=0}^{N-1} x(n) e^{-j(2\pi/N)kn}$, устраняющие свойственную ДПФ вычислительную избыточность. Они основываются на свойствах комплексной экспоненты $e^{-j(2\pi/N)kn}$, для удобства обозначаемой $W_N^{kn} (W_N = e^{-j(2\pi/N)kn})$ ее симметрии $W_N^{(N-k)n} = W_N^{(N-n)k} = (W_N^{kn})^*$ и периодичности $W_N^{(k+N)(n+N)} = W_N^{kn}$ с периодом, равным длине обрабатываемой реализации сигнала N (числу точек БПФ). В соответствии с последним свойством экспоненте $W_N^{pkn} = W_{N/p}^{kn}$ отвечает период N/p, где p – целые числа, на которые делится N. Использование данных свойств в алгоритмах БПФ исключает большое число повторяющихся при вычислении ДПФ операций. Общий принцип БПФ заключается в разбиении ДПФ исходной последовательности на ДПФ под последовательностей меньшей длины, вплоть до минимально возможной (равной основанию БПФ), через которые и вычисляется ДПФ исходной последовательности. Разбиение означает прореживание последовательностей во временной или в частотной области. В связи с этим различают БПФ с прореживанием по времени и БПФ с прореживанием по частоте. В отличие от ДПФ, БПФ может вычисляться только по определенному числу точек N, соответствующему целой степени его основания m: N = m L , где L – это число этапов прореживания: $L = \log_m N$. Чаще всего применяют БПФ по основанию 2 [1].

1.3 Выбор программного средства для разработки

1.3.1 Клиент-серверное приложение

Клиент-серверное приложение, в котором клиентом выступает браузер, а сервером – веб-сервер. Логика приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информации происходит по сети [2]. Основным преимуществом клиент-серверного приложения является кроссплатформенность, потому что при использовании данного приложения клиент не зависит от конкретной операционной системы. При использование данного подхода снижаются требования к устройствам, на которых установлен клиент, так как все вычисления происходят на стороне сервера.

1.3.2 Разработка клиентской части

Одной из составляющих приложения является клиентская часть, которая отображается пользователю. Данная часть реализует пользовательский интерфейс, формируя запросы к серверу и обрабатывает ответы на него. Основными компонентами клиентской части web-приложения является HTML, CSS. При реализации функционала web-приложения, для качественного и наглядного отображения данных, необходимо использовать библиотеки и фреймворки. В реализации мобильного приложения используется фреймворк Ionic.

1.3.2.1 HTML

HTML – специальный язык разметки, который используется для создания так называемых веб-документов. Аббревиатура HTML означает Hyper Text

Markup Language. Дословный перевод этих слов - Язык Разметки Гипертекста. Разберемся детальнее, что означают эти слова.

Гипертекст - это, вообще говоря, некоторый текст, который содержит ссылку на другой текст. В веб сфере эти ссылки являются ссылками на другие веб-документы.

Язык разметки - это набор синтаксических правил, с помощью которых можно упорядочено представить ту или иную информацию на экране. Важно отметить, что язык разметки не является языком программирования.

HTML-документ - это веб-документ, который был сформирован при помощи языка разметки HTML. По своей сути - это текстовый файл, в котором был использован язык разметки. Исходный код HTML-документа может быть просмотрен и отредактирован в любом текстовом редакторе, например, в Блокноте. Это одно из удобных свойств HTML-документа. Программа, которая позволяет отображать HTML-документ, на основе исходного кода разметки называется браузером.

Браузер интерпретирует HTML-документ, выстраивая его структуру и отображая ее в соответствии с инструкциями, включенными в этот файл (таблицы стилей, скрипты). Если разметка правильная, то в окне браузера будет отображена HTML-страница, содержащая HTML-элементы — заголовки, таблицы, изображения и т.д. [3].

Процесс интерпретации начинается прежде, чем веб-страница полностью загружена в браузер. Браузеры обрабатывают HTML-документы последовательно, с самого начала, при этом обрабатывая CSS и соотнося таблицы стилей с элементами страницы [3].

1.3.2.2 CSS

CSS (англ. Cascading Style Sheets — каскадные таблицы стилей) — формальный язык описания внешнего вида документа, написанного с использованием языка разметки [2].

Преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML, но может также применяться к любым XML-документам [2].

Каскадные таблицы стилей описывают правила форматирования элементов с помощью свойств и допустимых значений этих свойств. Для каждого элемента можно использовать ограниченный набор свойств, остальные свойства не будут оказывать на него никакого влияния [4].

CSS используется создателями веб-страниц для задания цветов, шрифтов, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц. Основной целью разработки CSS являлось разделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом. Кроме того, CSS позволяет представлять один и тот же документ в различных стилях или методах вывода, таких как экранное представление, печатное представление, чтение голосом (специальным голосовым браузером или программой чтения с экрана), или при выводе устройствами, использующими шрифт Брайля [2].

1.3.2.3 JavaScript

JavaScript - язык сценариев, с его помощью можно создавать интерактивные html-документы, производить вычисления, выполнять проверку допустимости данных без обращения к серверу [5].

Сценарий JavaScript не компилируется в бинарный код, а интерпретируется и обрабатывается интерпретатором, встроенным в браузер. Браузер начинает выполнять код JavaScript сразу же, как только обнаружит его на странице. Каждая строка кода выполняется последовательно, переход к следующей строке производится только после выполнения предыдущей. Если в документе содержится несколько сценариев, то они будут исполняться в порядке их расположения в документе [5].

Возможности JavaScript:

- изменять страницу: писать текст, добавлять и удалять теги, менять стили элементов;
- реагировать на события: скрипт имеет возможно ждать выполнения действия (клика мышки) и реагировать на это выполнением функции;
- Выполнять запросы к серверу и загружать данные без перезагрузки страницы;
- устанавливать и считывать cookie, проверять данные, выводить сообщения и многое другое.

JavaScript - объектно-ориентированный язык. Следует ознакомиться с тремя терминами:

- объекты - говоря простым языком, объект (object) - это какой-либо предмет (окно браузера, формы и их части, например кнопки и текстовые окна). В JavaScript также имеется собственная группа встроенных объектов, к которым относятся массивы, данные и т.д.;
- метод (method) - это действия, которые может выполнять объект. Примерами методов являются открытие и закрытие окон, нажатие кнопок. Здесь

речь идет о трех методах: open (), close () и click ();

- свойства - у всех объектов имеются свойства (properties).

Практически невозможно представить себе информационную панель без диаграмм и графиков. Они быстро и эффективно отображают сложные статистические данные.

JavaScript имеет множество библиотек для построения диаграмм/схем (и сводных таблиц). Эти библиотеки могут помочь в создании красивых и настраиваемых графиков для будущих проектов.

Использование JavaScript в web-приложении позволит динамично отображать данные.

1.3.2.4 TypeScript

TypeScript является обратно совместимым с JavaScript и компилируется в последний. Фактически, после компиляции программу на TypeScript можно выполнять в любом современном браузере или использовать совместно с серверной платформой Node.js. Код экспериментального компилятора, транслирующего TypeScript в JavaScript, распространяется под лицензией Apache. Его разработка ведётся в публичном репозитории через сервис GitHub.

TypeScript отличается от JavaScript возможностью явного статического назначения типов, поддержкой использования полноценных классов (как в традиционных объектно-ориентированных языках), а также поддержкой подключения модулей, что призвано повысить скорость разработки, облегчить читаемость, рефакторинг и повторное использования кода, помочь осуществлять поиск ошибок на этапе разработки и компиляции, и, возможно, ускорить выполнение программ.

Планируется, что в силу полной обратной совместимости адаптация существующих приложений на новый язык программирования может происходить поэтапно, путём постепенного определения типов.

На момент релиза представлены файлы для восприятия расширенного синтаксиса TypeScript для Vim и Emacs, а также плагин для MS Visual Studio.

Одновременно с выходом спецификации разработчики подготовили файлы с декларациями статических типов для некоторых популярных JavaScript-библиотек, среди которых jQuery.

1.3.2.5 Библиотека для построения графиков Chart.js

JavaScript-библиотека Chart.js позволяет генерировать на стороне клиента привлекательные графики и диаграммы с использованием средств HTML5 (canvas). Библиотека поддерживает создание линейных графиков, столбцовых, круговых и радиальных диаграмм. Chart.js поддерживает использование анимированных эффектов.

Большим достоинством Chart.js является ее небольшой размер и отсутствие каких-либо зависимостей. К тому же имеется возможность ещё уменьшить размер библиотеки путем включения в нее только тех модулей, которые необходимы в конкретном случае. К примеру, если нужно создать только диаграмму кругового типа (pie chart), то можно подключить ядро библиотеки Chart.js и модуль, с помощью которого создаются диаграммы подобного типа. Таким образом, будет уменьшен общий размер библиотеки Chart.js и увеличена скорость загрузки страницы в целом. Другим достоинством библиотеки является адаптивность диаграммы, позволяющая изменять свой размер при изменении размеров окна браузера.

1.3.2.6 Фреймворк Bootstrap 3

Популярный фреймворк, содержащий свободный набор инструментов для разработки адаптивных сайтов и мобильных web-проектов. Включает в себя HTML и CSS шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения [2].

Основные преимущества Bootstrap 3 [2]:

- экономия времени — Bootstrap позволяет сэкономить время и усилия, используя шаблоны дизайна и классы;
- высокая скорость — динамичные макеты Bootstrap масштабируются на разные устройства и разрешения экрана без каких-либо изменений в разметке;
- гармоничный дизайн — все компоненты платформы Bootstrap используют единый стиль и шаблоны с помощью центральной библиотеки.

Дизайн и макеты веб-страниц согласуются друг с другом;

- простота в использовании — платформа проста в использовании, пользователь с базовыми знаниями HTML и CSS может начать разработку с Twitter Bootstrap;
- открытое программное обеспечение — особенность Twitter Bootstrap, которая предполагает удобство использования, посредством открытости исходных кодов и бесплатной загрузки.

Bootstrap спроектирован для лучшей работы в новых браузерах, то есть старые браузеры не всегда могут правильно отображать стили, хотя полностью функциональны в визуализации определенных компонентов [7].

Далее приведена таблица поддержка браузерами Bootstrap 3.

Таблица 1 - Поддержка Bootstrap 3 браузерами

	Chrome	**Firefox**	**Internet Explorer**	**Opera**	**Safari**
Android	+	-	N/A	-	N/A
iOS	+	N/A	N/A	-	+
MAC OS X	+	+	N/A	+	+
Windows	+	+	+	+	-

Основные инструменты Bootstrap 3:

- сетки - заранее заданные размеры колонок, которые можно сразу же использовать;
- шаблоны — фиксированный или резиновый шаблон документа;
- типографика — описания шрифтов, определение некоторых классов для шрифтов, таких как код, цитаты и т. п.;
- медиа — представляет некоторое управление изображениями и видео;
- таблицы — средства оформления таблиц, вплоть до добавления функциональности сортировки;
- формы — классы для оформления форм и некоторых событий, происходящих с ними;
- навигация — классы оформления для табов, вкладок, страничности, меню и тулбара;
- алерты — оформление диалоговых окон, подсказок и всплывающих окон.

Использование Bootstrap 3 позволит создать адаптивный сайт, который будет хорошо просматриваемый на всех устройствах.

1.3.2.7 Фреймворк Vue.js

Vue – это прогрессивный фреймворк для создания пользовательских интерфейсов. В отличие от фреймворков-монолитов, Vue создан пригодным для постепенного внедрения. Его ядро в первую очередь решает задачи уровня представления (view), что упрощает интеграцию с другими библиотеками и существующими проектами. С другой стороны, Vue полностью подходит и для создания сложных одностраничных приложений (SPA, Single-Page Applications), если использовать его совместно с современными инструментами и дополнительными библиотеками [8].

1.3.3 Разработка серверной части

Важной частью веб-приложения и мобильного приложения является серверная часть. Программирование серверной части необходимо для динамичного отображения различных данных, располагающееся на сервере и отсылаемые клиенту.

1.3.3.1 Node.js

Node.js - программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API, подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода.

Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные

приложения (при помощи NW.js, AppJS или Electron для Linux, Windows и Mac OS) и даже программировать микроконтроллеры (например, tessel и espruino).

В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом [2].

1.3.3.2 Express.js

Express - это минималистичный и гибкий web-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и web-приложений. Автор фреймворка, TJ Holowaychuk, описывает его как созданный на основе написанного на языке Ruby каркаса Sinatra, подразумевая, что он минималистичен и включает большое число подключаемых плагинов. Express может являться backend'ом для программного стека MEAN, вместе с базой данных MongoDB и каркасом AngularJS для frontend'a.

1.3.3.3 Выбор средств разработки серверной части

Node.js – платформа, которая выводит язык JavaScript за пределы браузера и позволяет использовать его в серверных приложениях. В основе чплатформы лежит исключительно быстрый движок JavaScript, заимствованный из браузера Chrome, к которому добавлена быстрая и надежная библиотека асинхронного сетевого ввода/вывода. Основной упор в Node.js делается на создании высокопроизводительных, хорошо масштабируемых клиентских и серверных приложений.

1.3.4 Анализ технологий мобильных приложений

Native (с англ. Native - "родной") приложения – это те программные продукты, которые разрабатывались под конкретную операционную систему.

Например, если программа была написана под известную многим операционную систему Android, то она будет работать только на ней. У каждой операционной системы есть свой набор основных процедур (API), с помощью которого она осуществляет взаимодействие с разнообразными прикладными программами [1]. API (application programming interface) – набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) или операционной системой для использования во внешних программных продуктах [2].

Нативные приложения разрабатываются с конкретным API, а, следовательно, с конкретной операционной системой.

Преимущества нативного мобильного приложения:

- позволит поддерживать высокую производительность;
- позволит обрабатывать большое количество данных на стороне клиента;
- гибкость в реализации – нативная разработка может использовать все возможности мобильной операционной системы;
- экономичность расходования аккумулятора;
- использование платежных механизмов, предоставляемые магазинами приложений, например, Play Market для ОС Android.

Одним из главных недостатков нативного мобильного приложения можно выделить трудоемкость реализации проектов что, влияет на конечную стоимость продукта.

1.3.4.1 Кроссплатформенные приложения

Кроссплатформенное программное обеспечение — программное обеспечение, работающее более чем на одной аппаратной платформе и/или операционной системе [2].

Кроссплатформенные приложения – создаются на языке разметки HTML и стилей CSS, а так же JavaScript. Данные приложения пишутся одновременно для всех платформ и адаптивны к большинству устройств. Большинство разработчиков при разработке кроссплатформенных приложений пользуются фреймворком PhoneGap. Данный фрейморк позволяет получить доступ к аппаратным и программным возможностям платформы.

PhoneGap — бесплатный open-source, созданный Nitobi Software. Позволяет создать приложения для мобильных устройств используя JavaScript, HTML5 и CSS3, без необходимости знания «родных» языков программирования (например, Objective-C, Java), под все мобильные операционные системы такие как Android, iOS и так далее. Готовое приложение компилируется в виде установочных пакетов для каждой мобильной операционной системы [2].

Преимущества кроссплатформенной разработки:

- Экономия бюджета проекта, так как используется одна технология и один набор графики, что позволяет снизить количество рабочих часов;
- Время разработки – отсутствие уникальных элементов интерфейспозволяет значительно снизить сроки разработки.

Недостатки кроссплатформенной разработки:

- Не используются уникальные особенности платформ;
- Медленная работа приложения;
- Приложению необходим постоянный доступ в интернет, чтобы загружать контент.

1.3.4.2 Гибридный подход в разработке приложений

Гибридное приложение сочетает в себе некоторые функции, что имеет нативное и веб-приложение.

Достоинства гибридного приложения:

- кроссплатформенность. Сделав одно приложение, можно экспортировать его под любую операционную систему – iOS, Android, Windows Phone, BlackBerry;
- доступная стоимость разработки – в разы меньше, чем при нативном подходе;
- доступ к основным данным мобильного устройства: GPS, камера, телефонная книга и т.д.;
- возможность распространять приложение через официальные магазины приложений.

Недостатки гибридной разработки:

- Сниженная скорость работы в отличие от нативных приложений.
- Проблематичная верстка адаптивного дизайна. Несмотря на наличие различных веб-фреймворков для построения приложений

Ограниченное представление визуальных и графических элементов, в частности, анимации.

1.3.5 Выбор фреймворка для создания мобильного приложения

1.3.5.1 React Native

React Native — это JavaScript фреймворк, разработанный компанией Facebook для создания нативных мобильных приложений с использованием JavaScript. История React Native началась в 2013 году с внутреннего проекта хакатона компании Facebook. Впервые публично о React Native рассказали на React.js Conf в январе 2015 года, уже в марте 2015 на F8 объявили, что React Native доступен на GitHub. Фреймворк построен на основе библиотеки ReactJS. React Native использует JavaScript API поверх нативных компонентов. При создании приложения, пишется JavaScript код, который работает с нативными компонентами операционной системы. То есть, React Native использует те же

самые фундаментальные стандартные блоки UI что и обычные приложения iOS и Android, не используя при этом ни браузер, ни WebView/UIWebView.

Ниже представлена архитектура приложения на React Native [9].

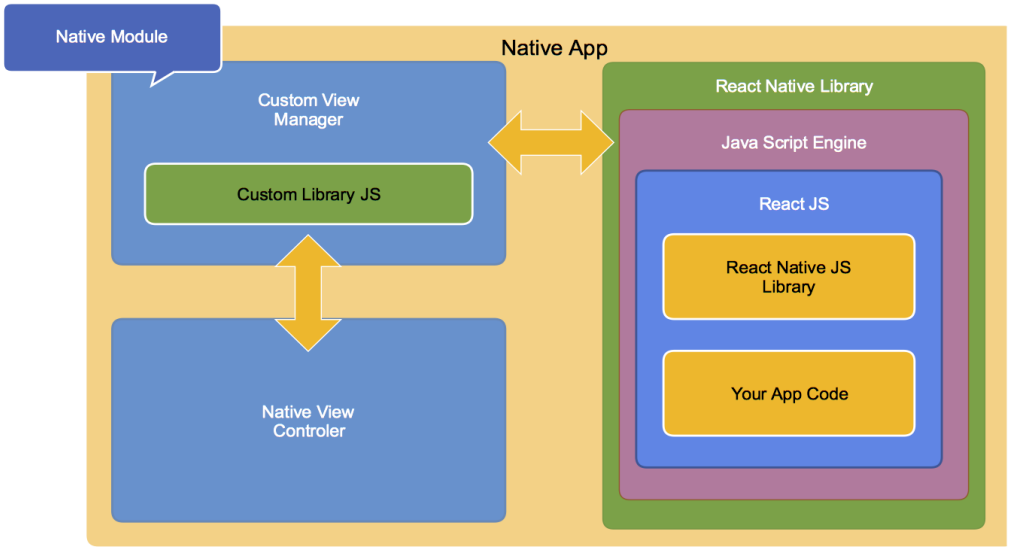

Рисунок 4 - Архитектура приложения на React Native

1.3.5.2 Ionic 3

Ionic Framework— это фреймворк для создания гибридных мобильных приложений. В его состав входит набор JavaScript и CSS компонент, созданных на основе AngularJS, SASS и Apache Cordova. История этого SDK начинается в 2013 году, когда компания Drifty Co. решила создать собственную инфраструктуру для написания гибридных приложений, которая будет ориентирована на производительность и будет построена с использованием современных веб-стандартов. Релиз Ionic 1 – состоялся в мае 2015 года. В 2016 году была выпущена версия 2. Основное отличия второго релиза от первого – это переход с Angular 1.x на Angular 2.x. По своей сути, Ionic Framework — это дополнение над очень популярным фреймворком Apache Cordova, но со своим мощным CLI (Command Line Interface) и объемной документацией.

Следуя принципам Apache Cordova, приложения на Ionic Framework — это гибридные HTML приложения. Такие приложения на телефоне выполняются в специальной оболочке (UIWebView для iOS и WebView для Android), которая позволяет показывать HTML и выполнять JavaScript. Соответственно при работе в приложении пользователь работает как бы в веб-браузере.

Ionic следует шаблону контроллера (View Controller), популяризированному в других фреймворках, например, Cocoa Touch. В шаблоне View Controller рассматриваются разные разделы интерфейса как дочерние представления вида или даже дочерние контроллеры вида, которые содержат другие виды. Затем контроллеры вида "приводят в действие" области просмотра, находящиеся внутри них, чтобы обеспечить взаимодействие и функциональность пользовательского интерфейса. Прекрасным примером может служить Tab Bar View Controller - контроллер вида вкладки в интерфейсе панели вкладок, который обрабатывает тапы по панели вкладок, с помощью которых происходит переключение между панелями, доступными для просмотра.

Ниже представлена архитектурная схема приложения Apache Cordova.

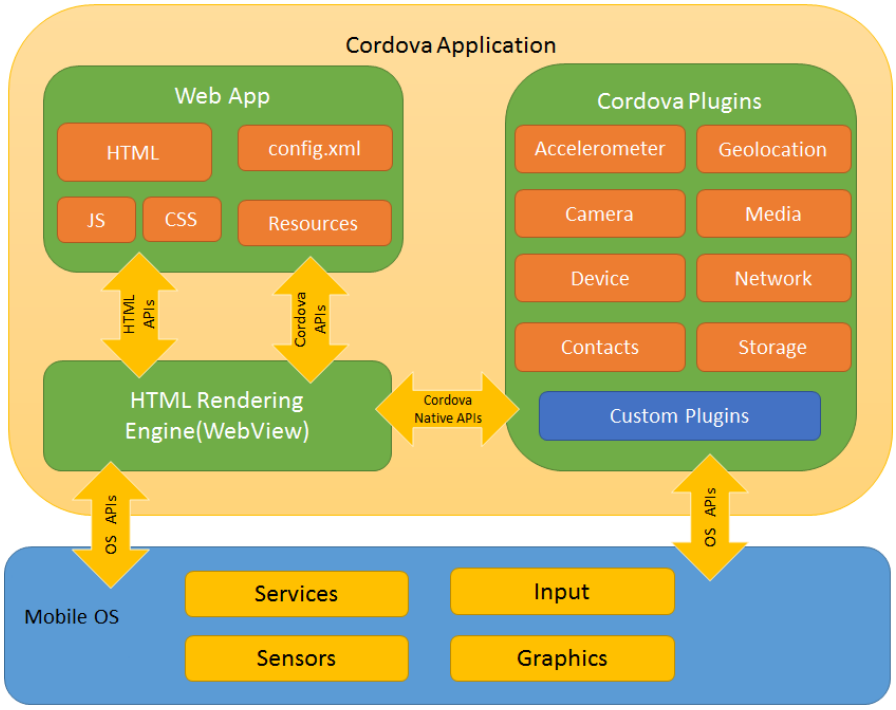

Рисунок 5 - Архитектура приложения на Apache Cordova

Apache Cordova является своеобразной прослойкой между пользовательским интерфейсом и ресурсами устройства. Какие-либо виджеты пользовательского интерфейса или Model-View фреймворков не входят в его состав. Если необходимо использовать UI-виджеты и/или Model-View фреймворк, то нужно выбрать и включить их в приложение самостоятельно, как ресурсы третьей стороны. Ionic 3 — этоs один из фреймворков, который предоставляет UI виджеты. MVC модель для него предоставляет Angular 4. Нативные функции телефона (например, камера, хранилище ключей, GPS

координаты) недоступны из веб-браузера. Поэтому для работы с ними используются плагины Apache Cordova. Кроме официальных плагинов есть ряд сторонних открытых плагинов.

1.3.5.3 Фреймворка для реализации мобильного приложения

Выбор фреймворка зависит от поставленных задач, конечной цели и многих других факторов. Использование Ionic 3 позволит быстро разработать прототип мобильного приложения. При использовании React Native можно столкнуться с рядом дополнительных задач. Например, необходимость работы с нативным кодом платформы ObjectiveC/Swift или Java.

Для реализации онлайн-стенда для мобильных устройств был выбран фреймворк Ionic 3, преимущества данного фреймворка представлены ниже.

- известный набор инструментов — для frontend разработчика набор используемых технологий является знакомым, это существенно сокращает время на выполнение поставленной задачи;
- быстрый старт — используя шаблоны приложений предоставленные Ionic, можно за короткое время создать прототип для показа заказчику;
- «write once, run anywhere» — написанное приложение работает на всех популярных платформах, это дает кросс-платформенность с незначительными изменениями кода.

2 Проектирование и разработка онлайн стенда

Лабораторный онлайн-стенд для цифровой обработки сигнала должен решать следующие задачи:

- генерация сигнал по заданным параметрам таким как частота дискретизации, длительность, частота сигнала, а также тип сигнала;
- проводить спектральный анализ сигнала для оценки влияния на спектр различных параметров;
- дискретная фильтрация сигнала;
- формировать произвольный сигнал;
- выполнять исследования с применением оконных функций.

2.1 Структурная схема онлайн-стенда

Структурная схема онлайн-стенда представлена на рисунке 6.

.

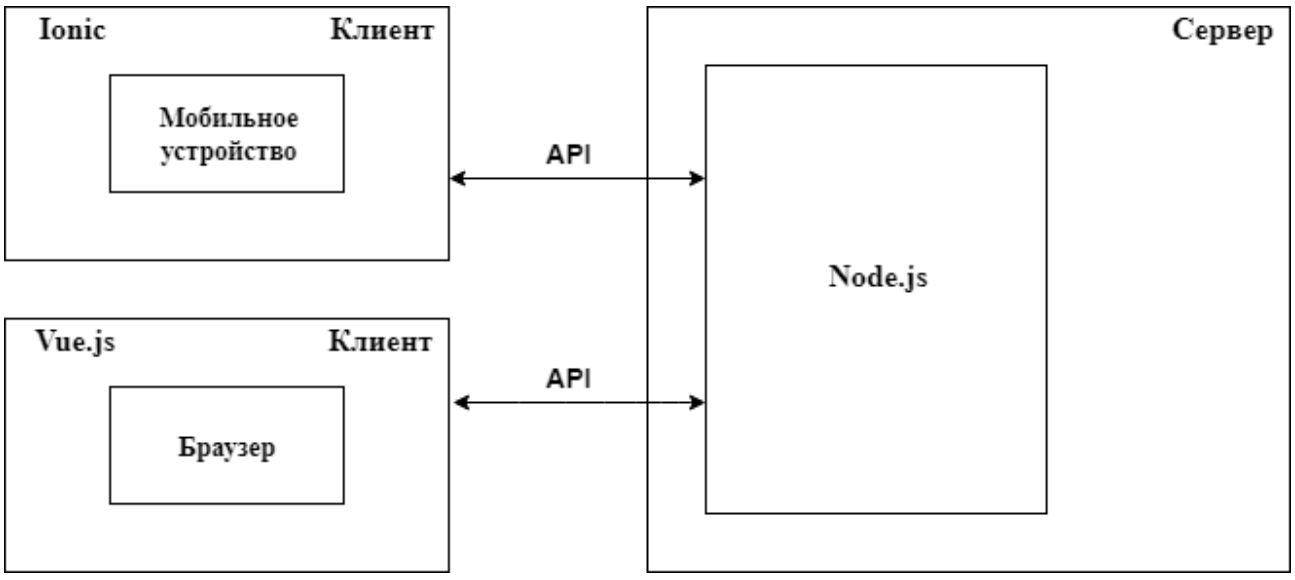

Рисунок 6 - Структурная схема онлайн-стенда

Структура системы разрабатывается с учётом задания, в котором определены основные характеристики и особенности ее реализации.

Онлайн-стенд для цифровой обработки сигнала состоит из клиентской и серверной части. Основные компоненты системы:

- фреймворк vue.js с помощью которого реализована клиентская часть web-приложения;
- мобильное приложение, реализованное с помощью фремворка Ionic 3;
- фреймворк express.js с помощью которого реализована серверная часть приложения, отвечающая за выполнение расчета математических функций.

2.2 Алгоритм работы системы

На рисунке 8 представлен алгоритм работы онлайн-стенда.

Для начала работы необходимо выбрать режим. Онлайн-стенд имеет пять режимов работы: генерация сигнала с заданными параметрами, спектральный анализ сигнала, применение оконной функции, дискретная фильтрация сигнала, генерация произвольного сигнала. Рассмотрим все режимы работы:

1. Генерация сигнала с заданными параметрами

Для начала работы необходимо выбрать тип сигнала. Системой предусмотрено три вида сигнала: синусоидальный, прямоугольный (меандр) и пилообразный. Далее выполняется ввод основных параметров таких как: частота дискретизации, длительность и частота сигнала. Для генерации необходимо нажать кнопку «сгенерировать».

2. Спектральный анализ сигнала

После ввода параметров нужно выбрать режим «анализ сигнала». Далее онлайн-стенд проведет вычисление быстрого преобразования Фурье на основе введенных параметров и построит спектральный график сигнала.

3. Применение оконной функции

Для построения графика с применением оконной функции так же необходим ввод основных параметров сигнала. Системой предусмотрено три вида оконных функций. Виды окон: прямоугольное окно, окно Хемминга, окно Наталла.

4. Дискретная фильтрация сигнала

Дискретная фильтрация сигнала происходит при условии выбора необходимых параметров сигнала. Далее необходимо указать импульсную характеристику в виде вектора отсчетов. После чего необходимо нажать кнопку «сгенерировать».

$$y(x) = \sum_{m=-\infty}^{\infty} x(m)h(k-m)$$

Рисунок 7 – Дискретная свертка

Импульсная характеристика – выходная реакция на единичный импульс.

5. Генерация произвольного сигнала

Для выполнения генерации произвольного сигнала необходимо указать функцию далее нажать кнопку «сгенерировать».

Доступные функции:

- sin – синус;
- cos – косинус;
- e – экспонента;
- sqrt – извлечения корня;
- deg – градусы;
- det – детерминант ;
- возведение в степень;
- сложение;
- вычитание;
- умножение;
- деление.

Примеры использования:

- sin(x)*cos(3x);
- sin(2x)+2^2;
- 2+3*1*2/2+sin(x)
- det([-1, 2; 3, 1])+cos(5x);
- sqrt(4)+sin(x);
- sin(45 deg) ^ 2 + cos(x).

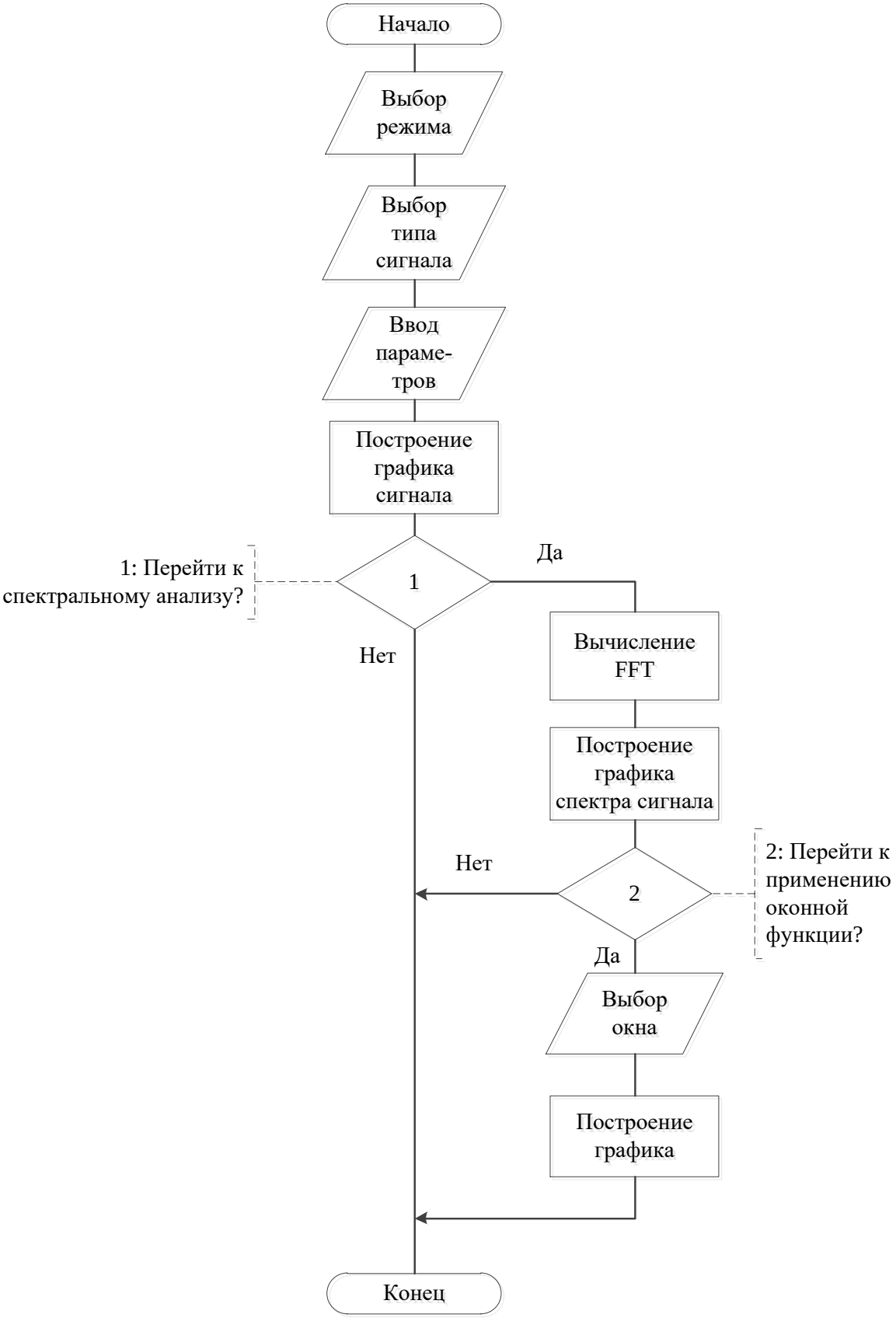

Рисунок 8 - Алгоритм работы системы

2.3 Технологии разработки онлайн-стенда

Лабораторный онлайн стенд состоит из пользовательской и серверной части.

Клиентская часть web-приложения строиться на языке разметки и гипертекста html, css – стилях и javascript-е. HTML нужен для отображения контента страницы: тексты, заголовки, текстовые блоки и т.д. CSS – необходим для описания внешнего вида документа, а именно это позиционирование элементов, установки границ блоков, размеры блоков, размер и цвет шрифта и т.д.

Клиентская часть для мобильных устройств реализована с помощью фреймворка Ionic, что позволяет реализовать кроссплатформенность приложения и оптимизировать время разработки приложения.

Серверная часть приложения реализовано с помощью фреймворка express.js, который позволяет использовать обширный набор функций.

2.3.1 Технологии разработки клиентской части web-приложения

2.3.1.1 Фреймворк Bootstrap 3

Bootstrap 3 – фреймворк позволяющий создавать адаптивные, современные, кросс-браузерные и стандартизированные интерфейсы. Данный Фреймворк отлично подходит для создания адаптивных шаблонов начиная от широкоэкранных мониторов и до экранов мобильных устройств с маленьким разрешением.

В файлах можно увидеть следующую структуру и содержание. Все файлы логически сгруппированы по общим свойствам и содержащие обе версии: минимизированную и компилированную. На рисунке 9 приведена структура файлов Bootstrap 3.

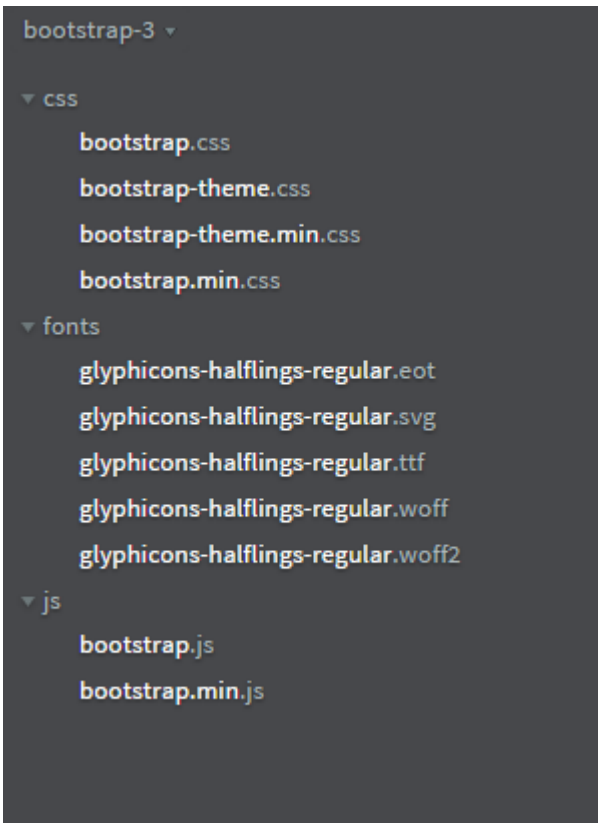

Рисунок 9 - Структура файлов Bootstrap 3

Для работы Фреймворка необходимо подключить файл стилей bootstrap, а также подключить javascript-файл, а также библиотеку JQuery.js:

CSS отделяет визуальное представление от структуры разметки HTML, точно так же, JQuery отделяет поведения от HTML. Например, вместо прямого указания на обработчик события нажатия кнопки, управление передаётся библиотеке JQuery, которая идентифицирует кнопки и затем преобразует его в обработчик события клика [2].

2.3.1.2 Фреймворк Vue.js

Vue.js работает «на уровне представления», то есть для создания клиентской части web-приложения и не используется для создания промежуточного обеспечения. Данный фреймворк легко интегрируется с

другими библиотеками. В данном случаи с bootstrap 3 и chart.js (библиотека для построения графиков). Vue.js содержит широкую функциональность для разработки клиентской части сайта.

Функции Vue.js:

- реактивные интерфейсы;
- декларативный рендеринг;
- связывание данных;
- директивы;
- компоненты;
- обработка событий;
- свойства;
- переходы и анимация CSS.

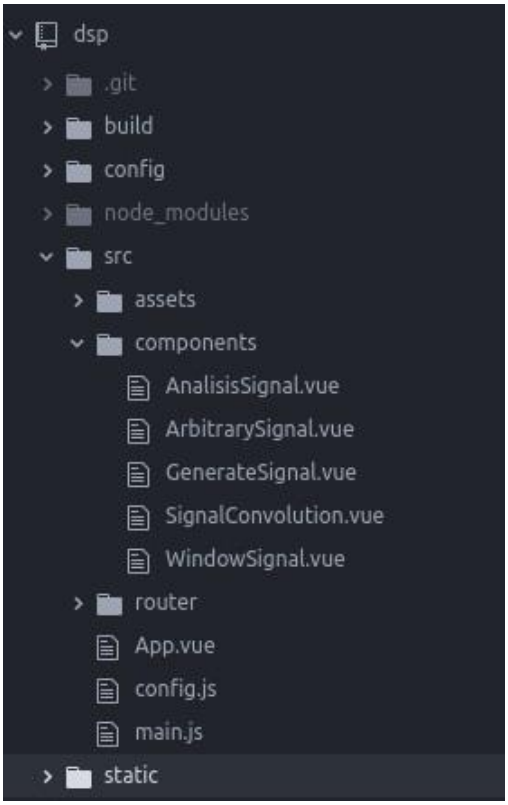

Рисунок 10 - Структура файлов Vue.js

2.3.2 Технологии разработки мобильного приложения

2.3.2.1 Ionic 3

Использование данного фреймворка позволит создать прототип кроссплатформенного мобильного приложения, что позволит сократить время потраченное на разработку, а также избежать работу с нативным кодом платформы таки как ObjectiveC/Swift или Java.

Ниже представлена структурная схема файлов фреймворка Ionic 3.

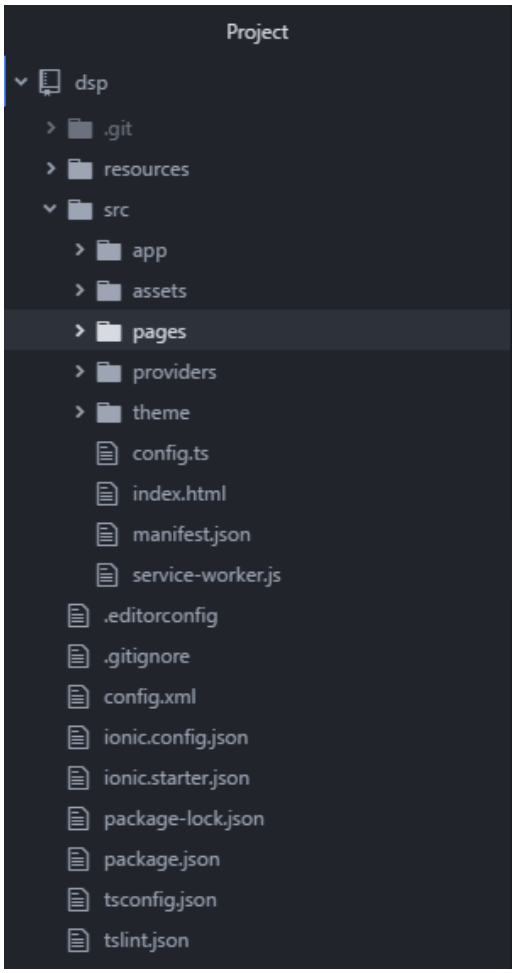

Рисунок 9 - Структура файлов мобильного приложения

2.3.3 Технологии разработки серверной части

2.3.3.1 Express.js

Express - это минималистичный и гибкий web-фреймворк для приложений Node.js. С помощью данного фреймворка реализованы все математические функции лабораторного онлайн стенда выполняющий роль web-сервера. К серверу приходит запрос с заданными параметрами сигнала и происходит обработка данных и формирование ответа для клиента.

На рисунке 10 структурная схема серверной части, реализованной с использованием express.js.

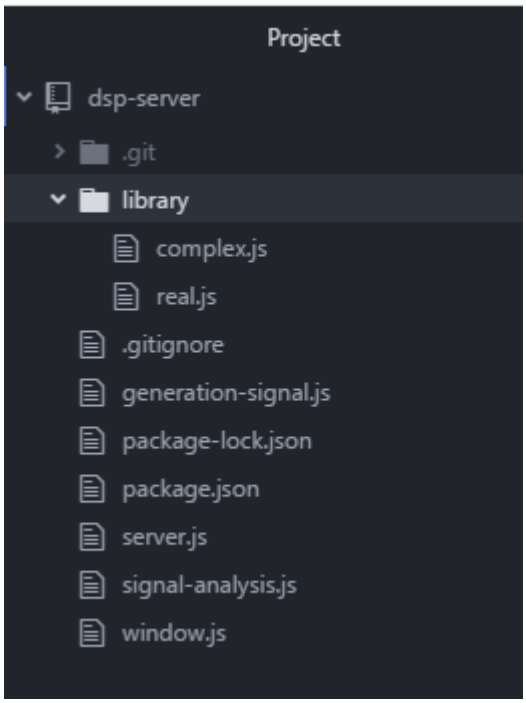

Рисунок 10 - Структура файлов серверной части приложения

2.4 Описание файлов системы

2.4.1 Описание файлов web-приложения

Онлайн-стенд состоит из нескольких частей. Описание файлов и информация, которая в них храниться приведено в таблице 2.

Таблица 2 - Описание файлов web-приложения

Имя файла	Назначение
App.vue	Основной компонент приложения
AnalisisSignal.vue	Компонент – анализ сигнала.
ArbitrarySignal.vue	Компонент – формирование произвольного сигнала.
GenerateSignal.vue	Компонент – генерация сигнала по заданным параметрам.
SignalConvolution.vue	Компонент – дискретная фильтрация.
WindowSignal.vue	Компонент – применение оконных функций.
router/index.js	Файл для настройки роутера приложения.
Jqplot.min.js	Пакет построения графиков и диаграмм, который выполнен в виде плагина для jQuery.
static/style.css	Файл стилей.
main.js	Файл инициализирующий основной компонент.

2.4.2 Описание файлов мобильного приложения

В таблице 3 представлено описание файлов компонента SignalGenerationPage, отвечающим за генерацию сигнала по заданным параметрам.

Таблица 3 – Описание файлов компонента мобильного приложения

Имя файла	Назначение
signal-generation.html	HTML-файл содержащий разметку страницы генерации сигналов по заданным параметрам
signal-generation.module.ts	Внутренний модуль компонента, необходим для логической группировки классов, интерфейсов, функций в один элемент.
signal-generation.scss	Файл стилей данной страницы.
signal-generation.ts	Содержит класс SignalGenerationPage для обработки данной страницы. Валидацию входных параметров и построение графики сигнала.

Структурна схема файлов других страниц аналогично с данной страницей.

2.4.3 Описание файлов серверной части приложения

Далее представлено описание файлов серверной части лабораторного онлайн стенда.

Таблица 4 – Описание файлов серверной части

Имя файла	Назначение
package.json	Файл, описывает зависимости верхнего уровня.
package-lock.json	Файл, представляющий слепок текущих зависимостей.
server.js	Основной файл сервера, отвечающий за дальнейший вызов функций в зависимости от адреса запроса.
generation-signal.js	Файл, отвечающий за расчет параметров сигнала по заданным параметрам.
signal-analysis.js	Файл, отвечающий за расчет параметров для частотного анализа сигнала.
window.js	Файл, отвечающий за расчет параметров для построения графика оконной функции
arbitrary-signal-analysis.js	Файл, отвечающий за дискретную фильтрацию сигнала
library\complex.js	Скрипт для вычисления Быстрого преобразования Фурье.

2.5 Описание работы онлайн-стенда

2.5.1 Описание работы web-приложения

Для построения и отображение графиков используется библиотека chart.js поддерживающая различные виды графиков.

Работа в любом из режимов, представленных в лабораторном онлайн стенде начинается с ввод необходимых параметров. Следующим этапом будет валидация введенных данных, то есть проверка на их корректность. Если указанные данные не верны, пользователь получит соответствующую ошибку. После чего выполняется запрос на сервер с введёнными параметрами. На сервере происходи обработка данных и формирование ответа для клиентской части. Ответ представляет собой массив значений для построения графика сигнала. Далее происходит указание наименование графика и параметров осей.

2.5.2 Описание работы мобильного приложения

В ходе работы был проведен анализ технологий, выделены их основные преимущества и недостатки. Исходя из этого был выбран фреймворк Ionic для реализации мобильного приложения.

Iomic 3 следует шаблону View – Controller. Controller отвечающий за формирование данных и View отвечающий за отображение данных пользователю.

Принцип работы мобильного приложения аналогичен с принципом работы web-приложения. Далее будут представлены этапы работы в режиме генерация сигналов по заданным параметрам:

1. ввод параметров сигнала, таких как частота дискретизации, чистота и длительность сигнала и амплитуда (см.приложение А).

2. проверка на корректность введённых данных, в случае неудачи пользователь получит ошибку;

3. далее происходит вызов функции getSignalParams описанной в провайдере GenarationSignalProvider. Данная функция формирует запрос к серверу передавая параметры введение на первом этапе (см.приложение А).

4. следующим этапом будет получение ответа от сервера и построение графика сигнала (см.приложение А).

2.5.3 Описание работы серверной части приложения

Серверная часть лабораторного онлайн стенда отвечает за выполнение математических функций. Принцип работы серверной части рассмотрим на режиме работы спектрального анализа сигнала. На сервер приходит запрос с адресом необходимым для вызова функции и параметрами необходимыми для выполнения данной операции. Функция отвечает за расчет дискретного преобразования Фурье, данный расчет осуществляется алгоритмом быстрого преобразования Фурье. После чего будет сформирован ответ для клиента представленный в виде массива (см.приложение Б).

2.6 Функционал онлайн-стенда

В процессе разработки онлайн стенда для цифровой обработки сигнала было необходимо реализовать следующие функции:

1. Генерация сигнала с заданными параметрами;

2. Вычисление дискретного преобразование Фурье и построение графика спектрального анализа;

3. Построение графика с применением оконной функции;

4. Генерация произвольного сигнала;
5. Дискретная фильтрация сигнала.
6. Генерация изображений

Первый пункт – генерация сигнала с заданными параметрами. На вход поступают данные введённые пользователем такие как: частота дискретизации, длительность сигнала, частота сигнала, амплитуда сигнала. Ввод входных данных для генерации сигнала представлен на рисунке 11.

Рисунок 11 - Ввод входных данных

Рассмотрим более подробно каждый из параметров:

1. Частота дискретизации - частота взятия отсчетов непрерывного во времени сигнала при его дискретизации. Измеряется в герцах. Чем выше частота дискретизации, тем более широкий спектр сигнала может быть представлен в дискретном сигнале [2].

2. Длительность сигнала определяет интервал времени, в котором данный сигнал существует [2].

3. Частота сигнала – характеризует количество повторений в единицу времени.

4. Амплитуда сигнала - максимальное значение смещения или изменения переменной величины от среднего значения при колебательном или волновом движении [2].

Так же пользователь должен выбрать вид графика сигнала как показано на рисунке 12.

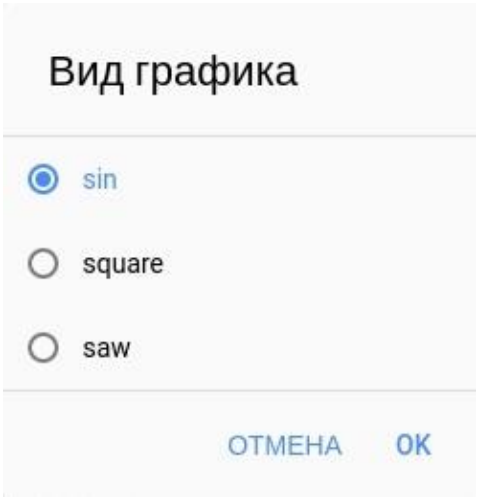

Рисунок 12 - Вид графика

Рассмотрим каждый вид:

1. Sin – синусоидальный сигнал. Синусоидальные сигналы распространены наиболее широко. Математическое выражение описывающее синусоидальный сигнал имеет вид: $s(t) = A\sin(2\pi ft + \varphi)$, где A - амплитуда сигнала, f - частота сигнала измеряемая в герцах, φ -фаза.

2. Square – прямоугольный сигнал. Прямоугольный сигнал содержит основную гармонику плюс бесконечное множество нечетных гармоник.

3. Saw – пилообразный сигнал. Пилообразный сигнал содержит основную гармонику плюс бесконечное множество четных гармоник.

Второй пункт – вычисление дискретного преобразование Фурье и построение графика спектрального анализа. Вычисление дискретное преобразование Фурье осуществляется алгоритмом быстрого преобразования Фурье, который минимизирует число математических операций, необходимых для его вычисления.

Третий пункт – построение графика сигнала с применением оконной функции. Список поддерживаемых приложением окон представлен на рисунке 13.

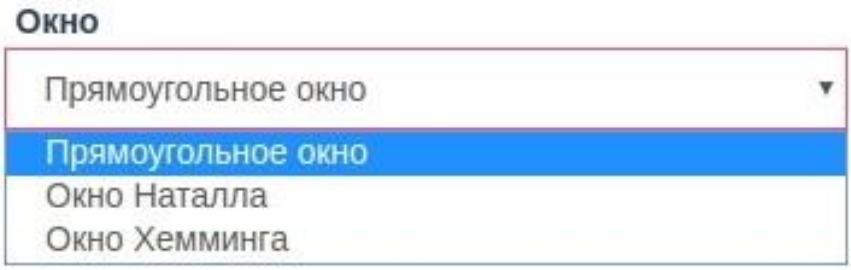

Рисунок 13 - Виды окон

Выражения для расчета оконных функций представлены на рисунке 14.

Наименование окна	Выражение в дискретном виде: $w(n),\ n=0\ldots N-1$	Примечание
Прямоугольное окно (rectangle window)	$w(n)=1$	Окно высокого разрешения минимальная ширина главного лепестка, но максимальный уровень боковых лепестков
Окно Хемминга (Hamming window)	$w(n)=0.54-0.46\cdot\cos\left(\frac{2\cdot\pi\cdot n}{N-1}\right)$	Окно высокого разрешения. Наилучшее окно при $K=2$
Окно Наталла (Nuttall window)	$w(n)=a_0-a_1\cdot\cos\left(\frac{2\cdot\pi\cdot n}{N-1}\right)+a_2\cdot\cos\left(\frac{4\cdot\pi\cdot n}{N-1}\right)-a_3\cdot\cos\left(\frac{6\cdot\pi\cdot n}{N-1}\right),$ $a_0=0.355768,\ a_1=0.487396,\ a_2=0.144232,\ a_3=0.012604$	Окно низкого разрешения

Рисунок 14 - Выражения для оконных функций

Четвертый пункт – генерация произвольного сигнала. Пример функции представлен на рисунке 15.

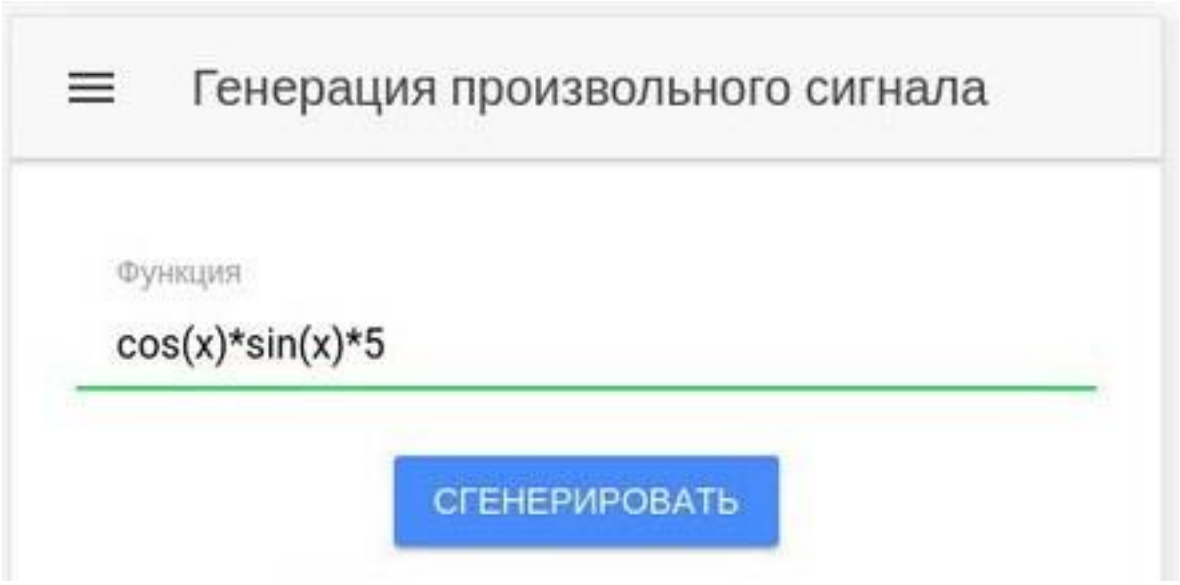

Рисунок 15 – Функция для генерации произвольного сигнала

Пятый пункт - дискретная фильтрация сигнала. Необходимо указать параметры сигнала. Далее пользователю будет предложено указать импульсную характеристику.

Рисунок 16 – Импульсная характеристика

Шестой пункт – полученные результаты возможно экспортировать в изображение.

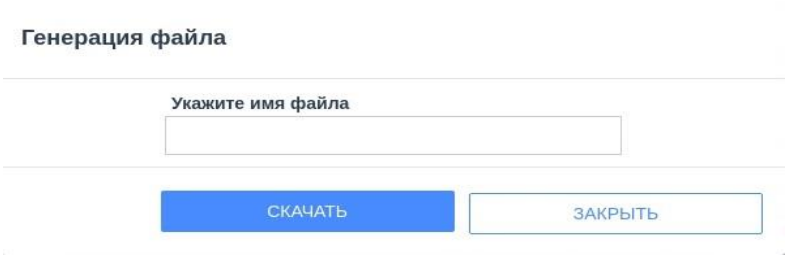

Рисунок 17 – Генерация файла

3 Тестирование онлайн-стенда

3.1 Общие принципы тестирования

При тестировании следует проверить соответствует ли разработанный онлайн-стенд заявленным требованиям. В ходе тестирования будет проверена работа функционала, например, генерация графика сигнала по заданным параметрам, построение графика спектрального анализа.

3.2.1 Тестирование web-приложения

3.2.1.1 Генерация сигнала по заданным параметрам

После того, как пользователь зашел на сайт ему предоставляется некоторый набор функций. Одной из таких функций является генерация сигнала по заданным параметрам. Пользователю необходимо ввести частоту дискретизации, частоту сигнала, амплитуду и длительность, а также указать вид графика. После чего следует нажать кнопку «Сгенерировать». Результат работы представлен на рисунке 18, 19.

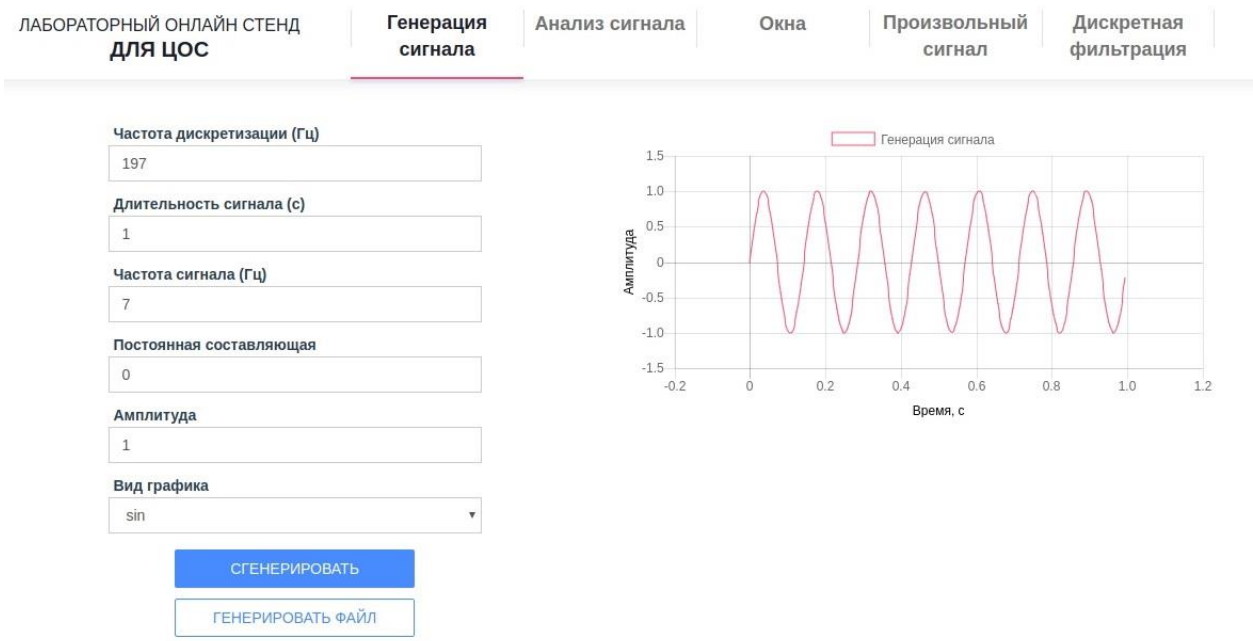

Рисунок 18 - График исходной функции синусоидального сигнала

При частоте дискретизации – 197 Гц, частоте сигнала – 7 Гц, типе сигнала – синусоидальный сигнал, получается результат, представленный на рисунке 18.

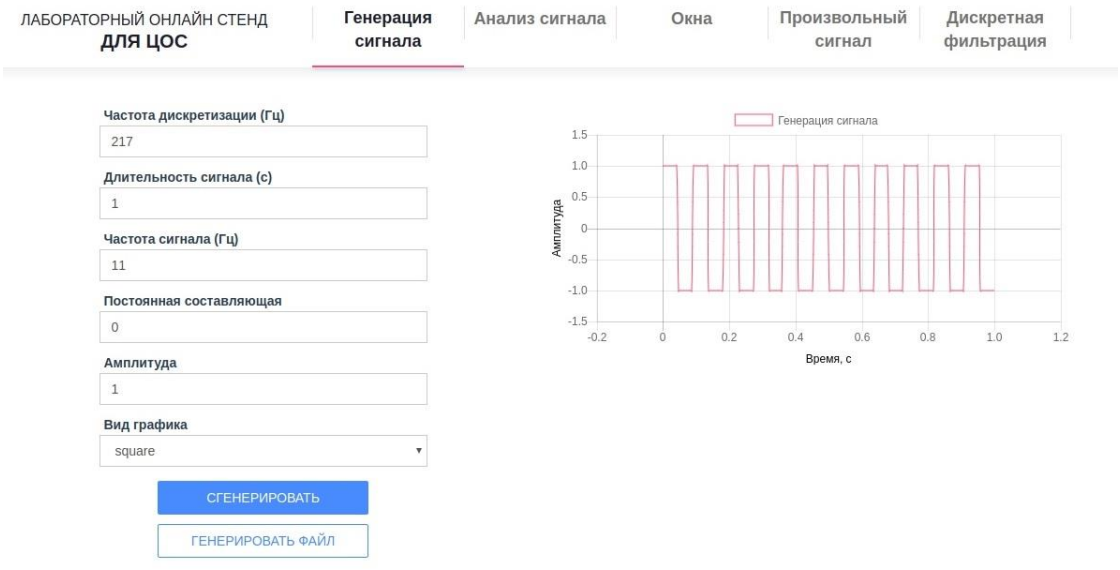

Рисунок 19 - График исходной функции прямоугольного сигнала

При частоте дискретизации – 217 Гц, частоте сигнала – 11 Гц, амплитуде сигнала – 1, длительности сигнала – 1, типе сигнала –прямоугольный сигнал, получается результат, представленный на рисунке 19.

3.2.1.2 Спектральный анализ сигнала

Ещё одной из функций, предоставляемых пользователю является спектральный анализ сигнала.

Ввод основных параметров представлен на рисунке 20.

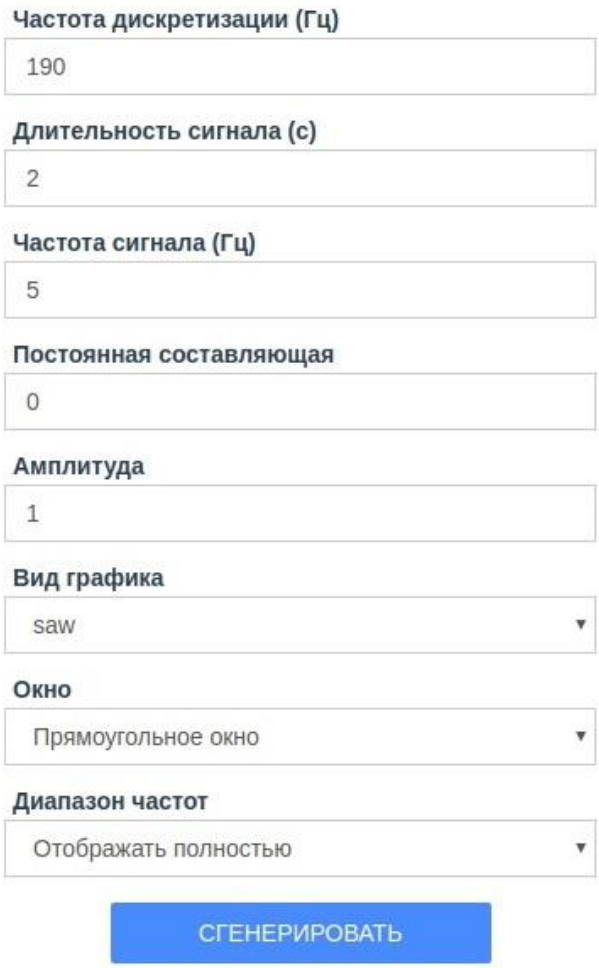

Рисунок 20 - Ввод параметров

После того, как был сгенерирован сигнал по заданным параметрам, пользователь переходит во вкладку анализ сигнала. Далее необходимо нажать кнопку «сгенерировать». В данной вкладке будет представлен график спектрального анализа сигнал. Результаты работы представлены на рисунке 21.

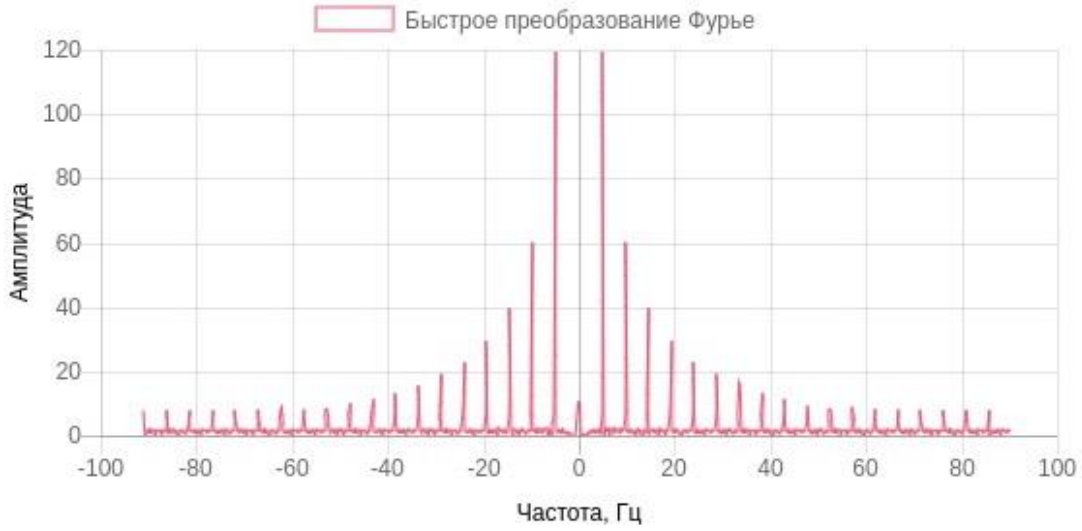

Рисунок 21 - Спектральный анализ пилообразного сигнала

При частоте дискретизации – 169 Гц, частоте сигнала – 3 Гц, амплитуде сигнала – 1, длительности сигнала – 3, типе сигнала –прямоугольный сигнал, стоится спектральный график сигнала, представленный на рисунке 20.

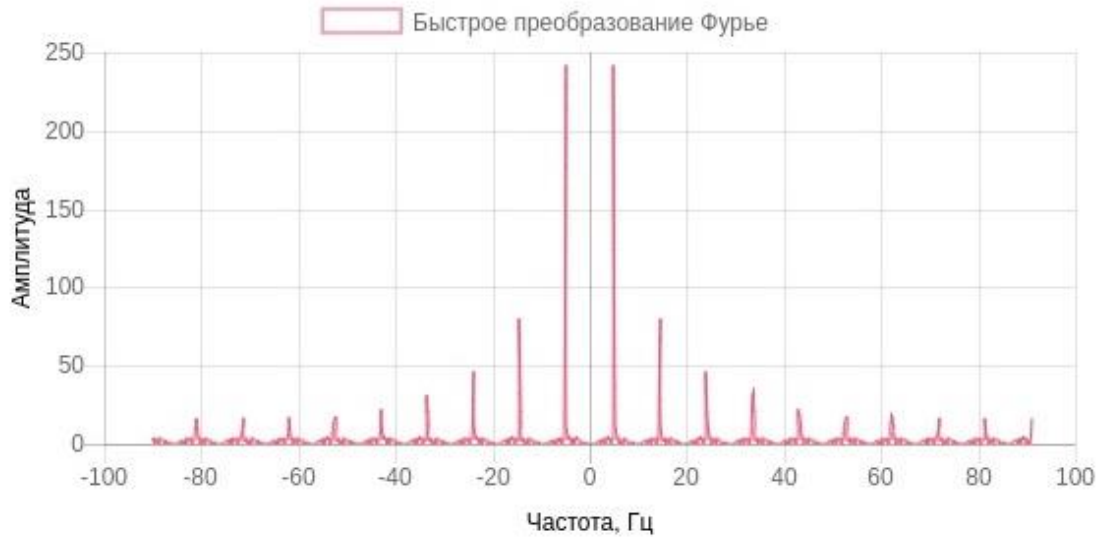

Рисунок 22 - Спектральный анализ прямоугольного сигнала

Если применить параметр диапазон частот, то график «Быстрое преобразование Фурье» будет иметь вид, представленный на рисунке 23, вместо графика представленного на рисунке 22.

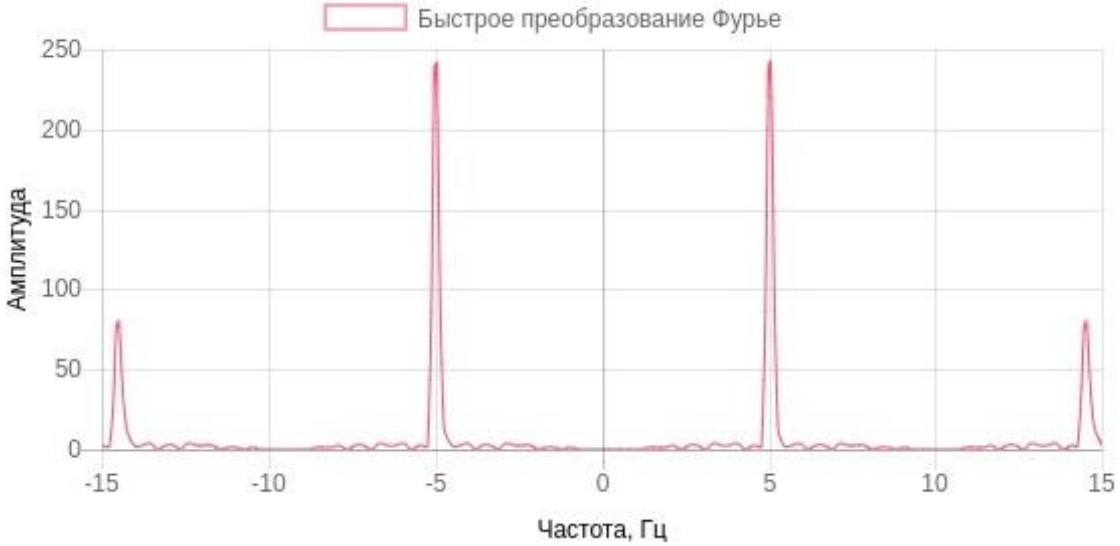

Рисунок 23 - БПФ прямоугольного сигнала с применением параметра диапазон частот 20%

3.2.1.3 Графика сигнала с применением оконной функции

Для того чтобы построить график сигнала с применением оконной функции пользователю необходимо перейти во вкладку «Окна» и выбрать вид окна. Системой предусмотрено три вида окна: прямоугольное окно, окно Хемминга, а также окно Наталла. После того как был выбран вид окна, необходимо нажать кнопку «сгенерировать». Пользователю будет представлен график оконной функции и результат примененной функции к сигналу с заданными параметрами.

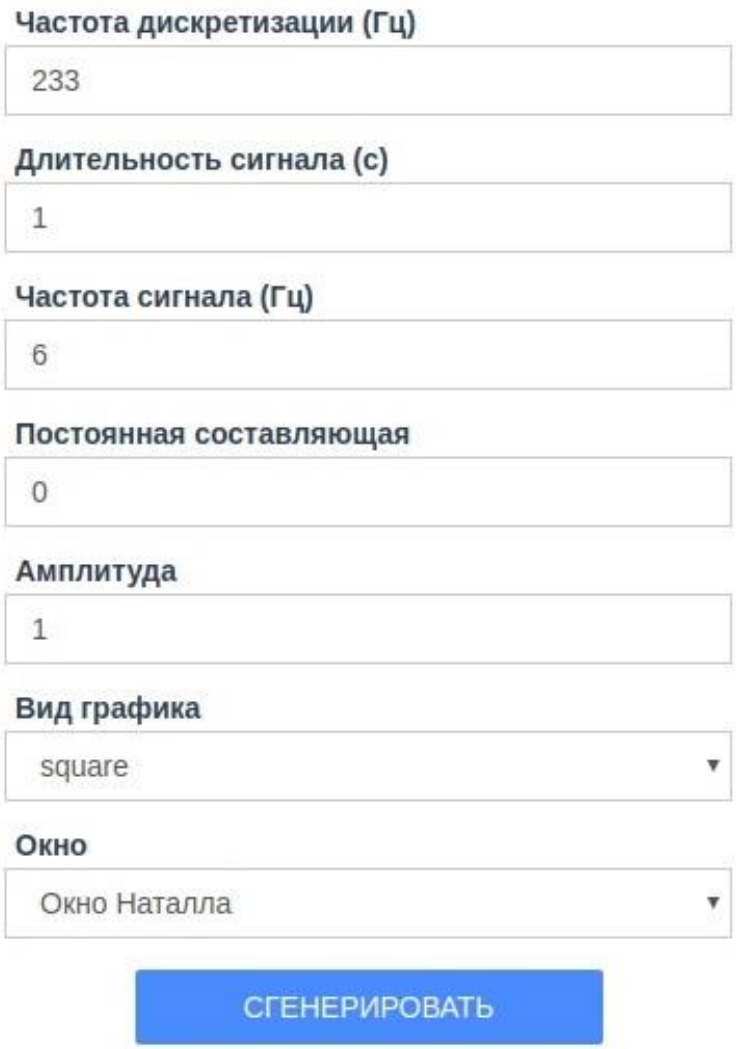

Рисунок 24 – Параметры сигнала для применения оконной функции

При частоте дискретизации – 233 Гц, частоте сигнала – 6 Гц, амплитуде сигнала – 1, длительности сигнала – 1, типе сигнала –прямоугольный сигнал. Вид окна – окно Наталла. График применения оконной функции представлен на рисунке 25.

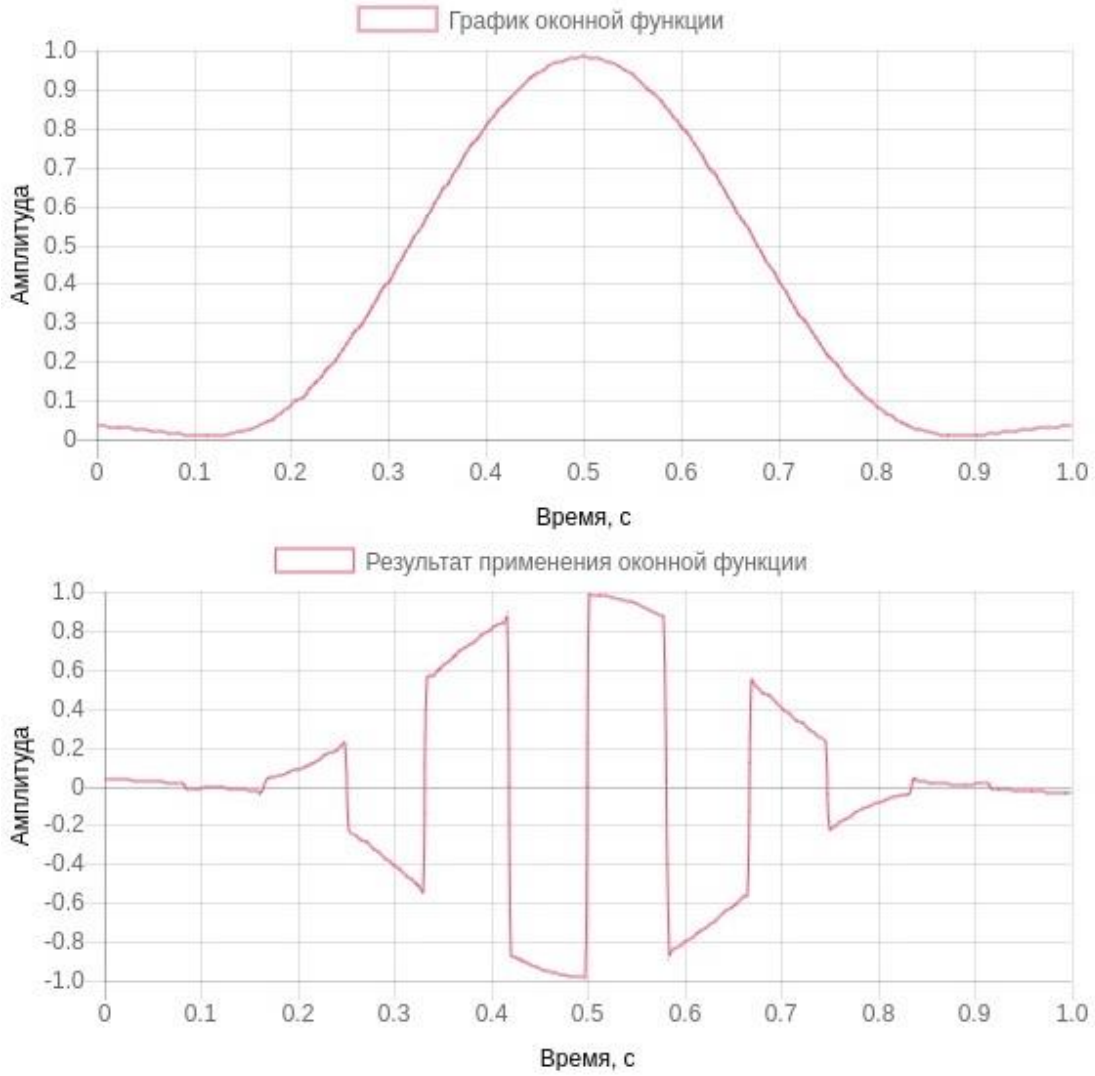

Рисунок 25 - График оконной функции Наталла и результат применения

Весь функционал онлайн-стенда работает исправно, графики функций строятся правильно.

3.2.1.4 Генерация произвольного сигнала

Необходимо указать функцию для генерации сигнала и временной интервал. Результаты работы представлены на рисунке 26.

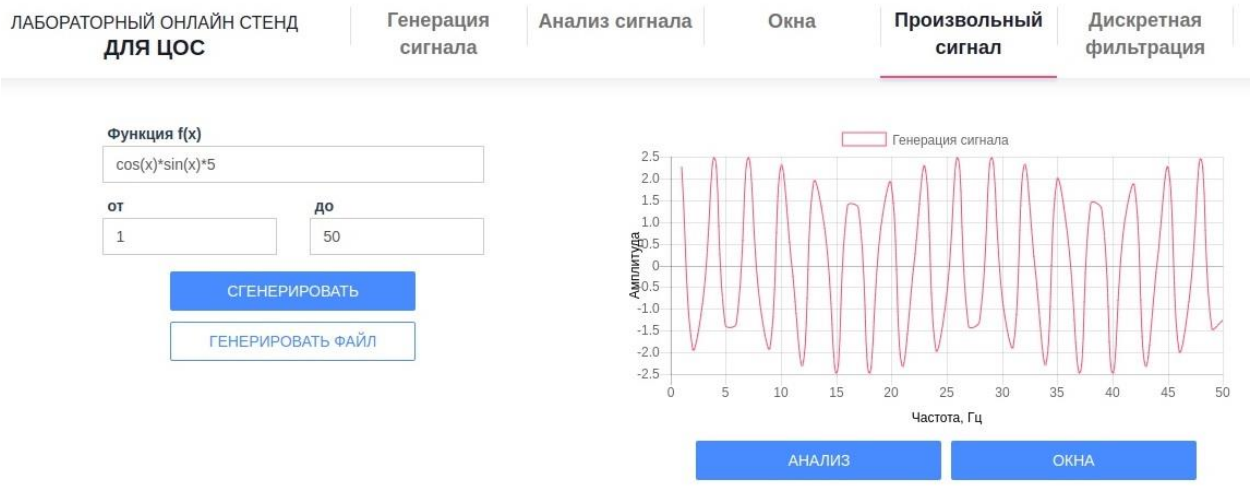

Рисунок 26 – Генерация произвольного сигнала

На основе полученных результатов, системой предусмотрена возможность провести анализ сигнала и выполнить исследование с применением оконных функций.

3.2.1.5 Дискретная фильтрация сигнала

Первым шагом необходимо указать параметры сигнала и выбрать вид графика. Вторым шагом необходимо указать импульсную характеристику. При частоте дискретизации – 233, частоте сигнала – 6, длительности и амплитуде сигнала – 1. Результат дискретной фильтрации представлен рисунке 27.

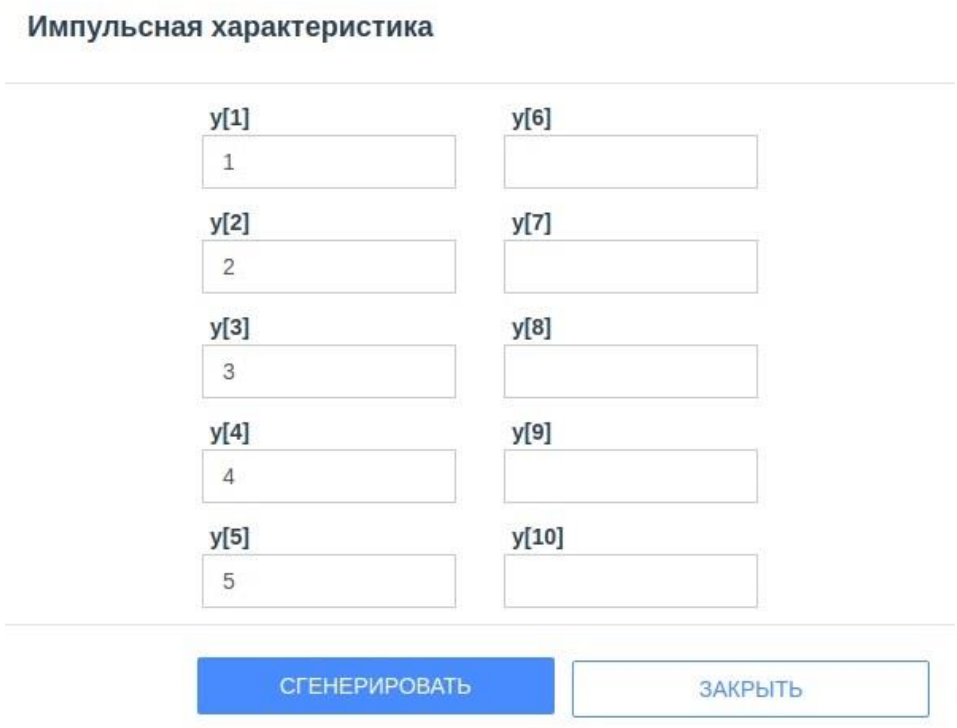

Рисунок 27 – Импульсная характеристика

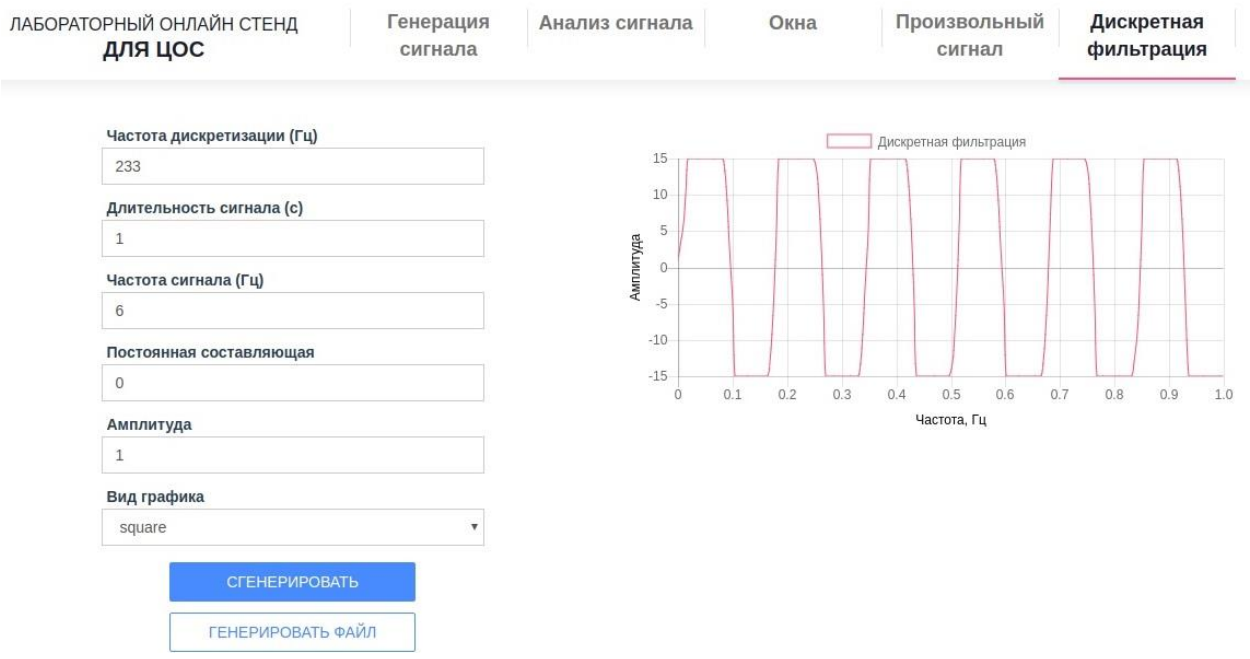

Рисунок 28 – Дискретная фильтрация

3.2.2 Тестирование мобильного приложения

3.2.2.1 Генерация сигнала по заданным параметрам

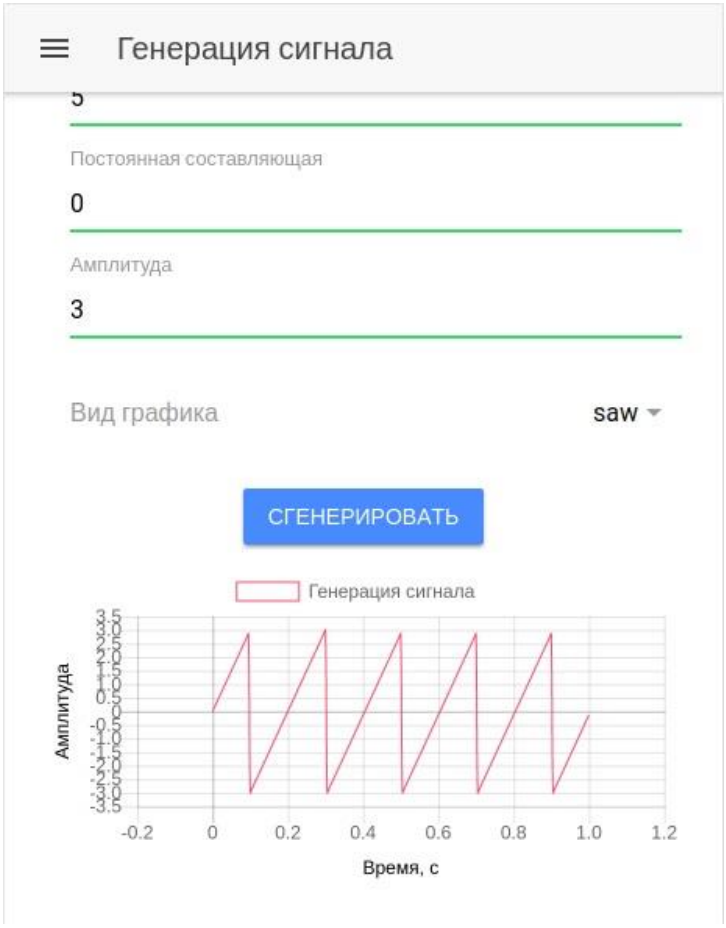

Рисунок 29 - График пилообразного сигнала

При частоте дискретизации - 220 Гц, длительности сигнала - 1 с, частоте - 5 Гц и амплитуде сигнала равной 3 будет сформирован сигнал, представленный на рисунке 29.

3.2.2.2 Спектральный анализ сигнала

При частоте дискретизации – 220 Гц, частоте сигнала – 3 Гц, амплитуде сигнала – 3, длительности сигнала – 1, типе сигнала –синусоидальный сигнал, генерируется спектральный график сигнала, представленный на рисунке 30.

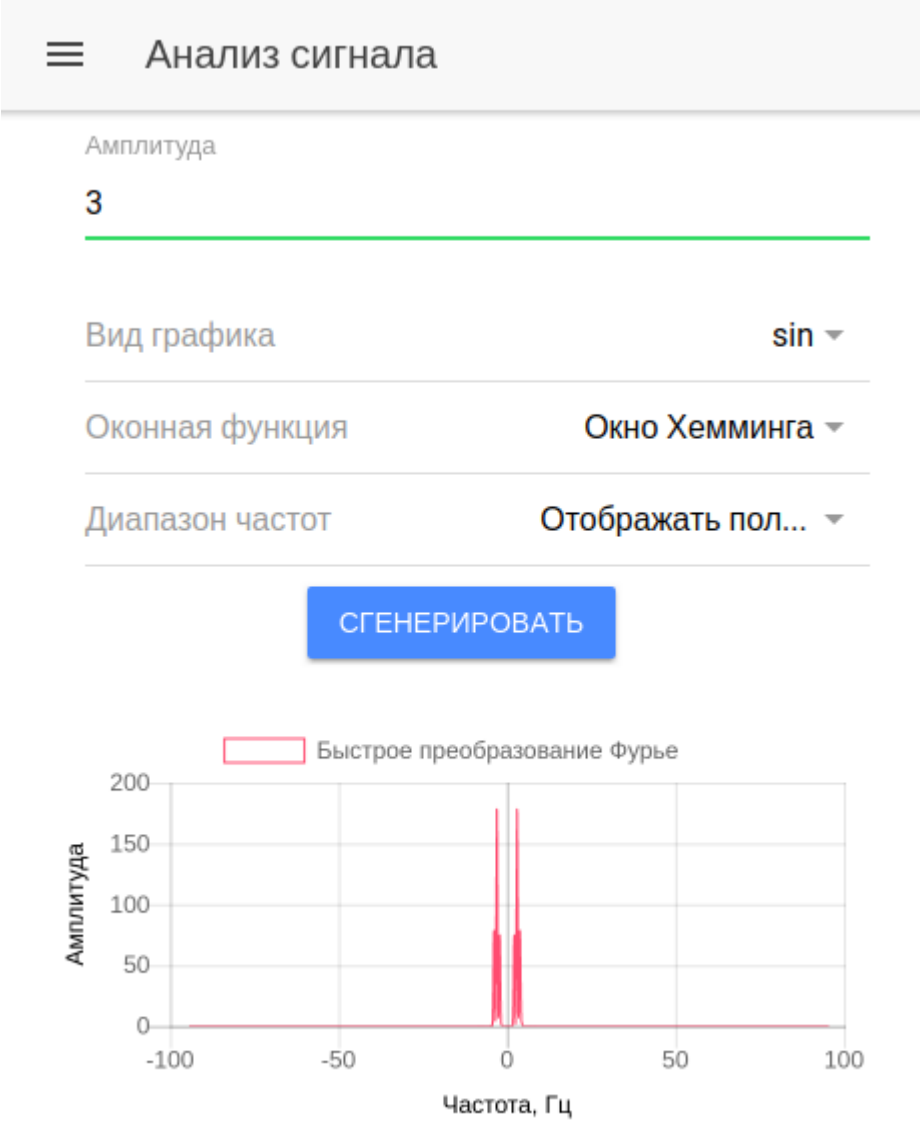

Рисунок 30 – Спектральный анализ сигнала

Если применить параметр «диапазон частот», то график «Быстрое преобразование Фурье» будет иметь вид, представленный на рисунке 31.

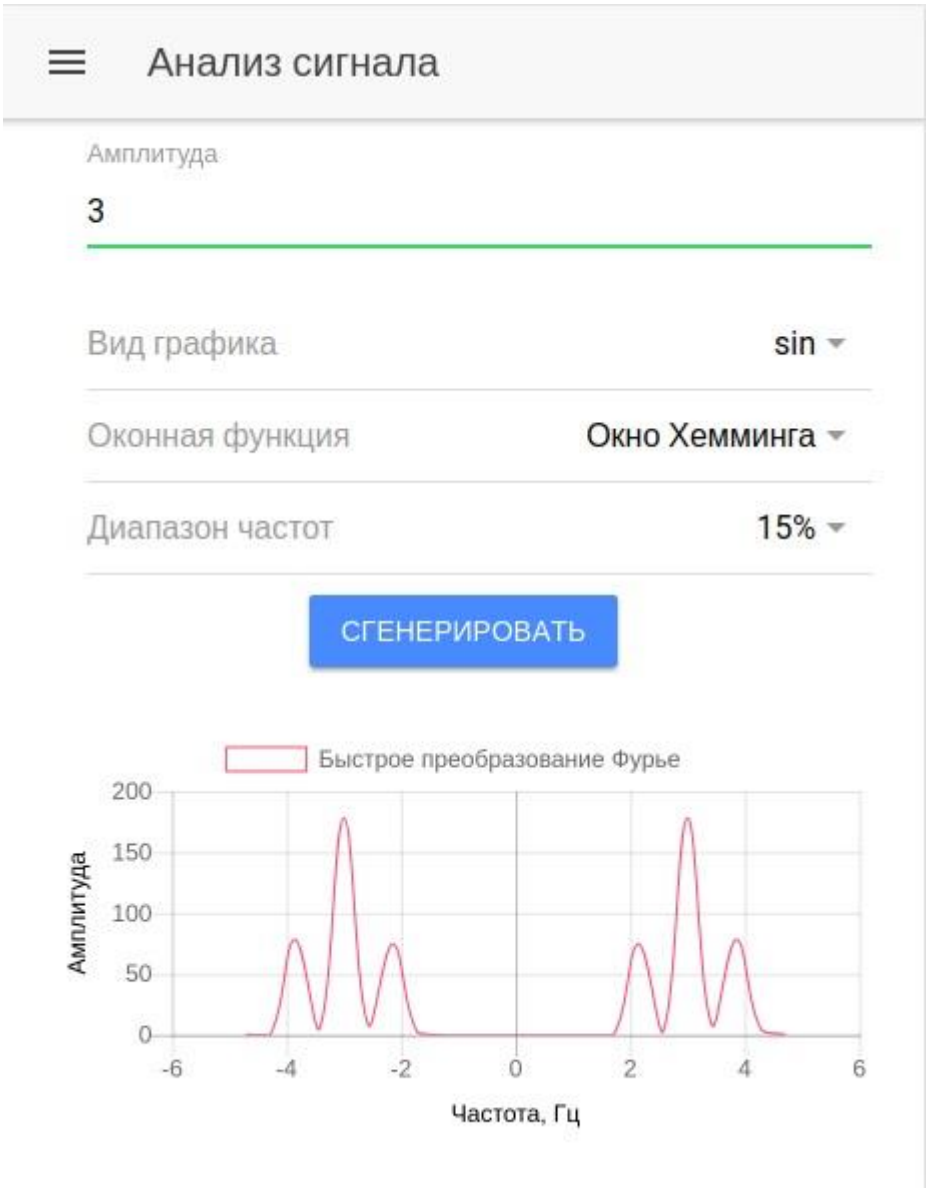

Рисунок 31 – БПФ прямоугольного сигнала с применением параметра диапазон частот 15%

3.2.2.3 Графика сигнала с применением оконной функции

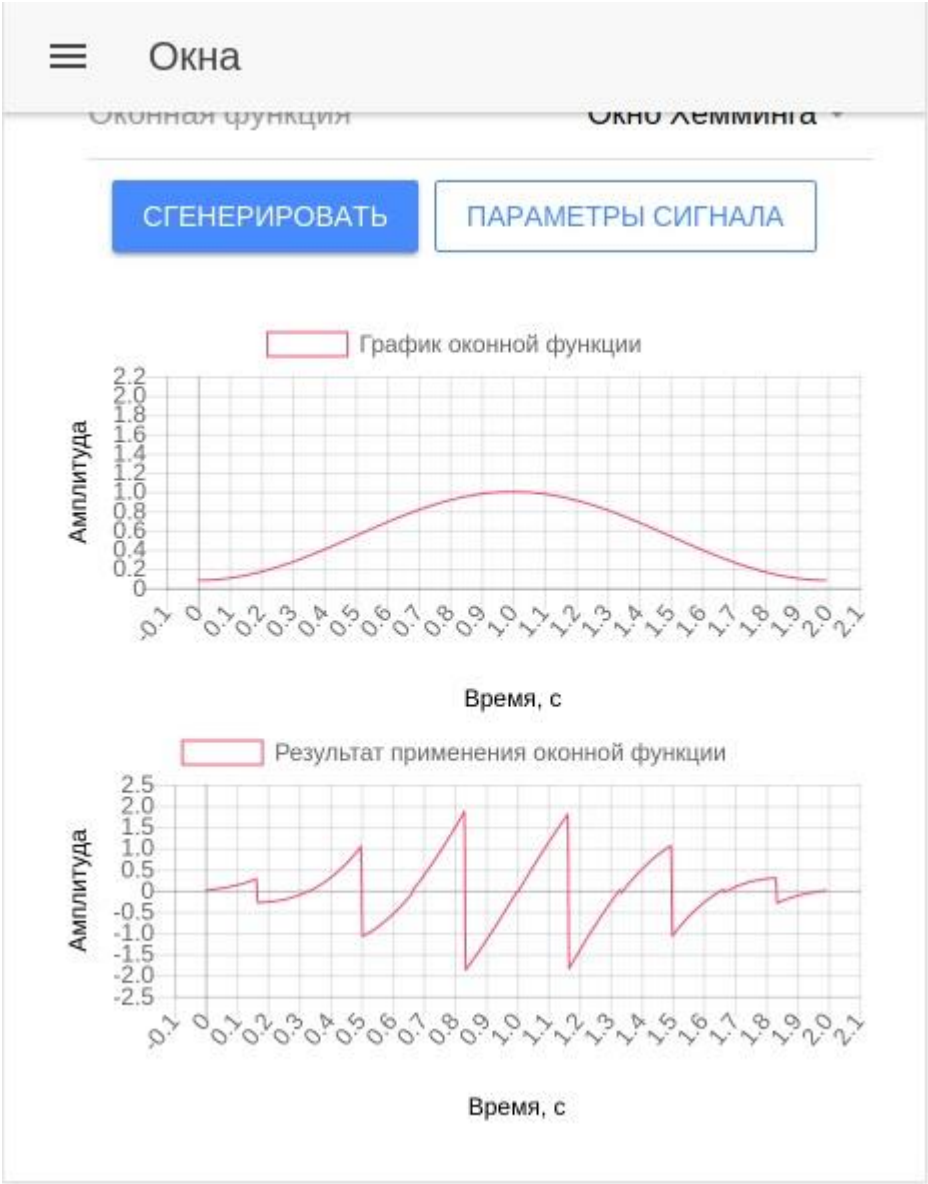

Рисунок 32 – Результат применения оконной функции

На рисунке 32 представлен результат применения оконной функции. При заданных параметрах частота дискретизации – 220 Гц, длительность сигнала – 1 с, частота сигнала – 3 и амплитуда 2.

3.2.2.4 Генерация произвольного сигнала

Для генерации произвольного сигнала необходимо указать функцию, как показано на рисунке 33.

Рисунок 33 – Генерация произвольного сигнала

3.2.2.5 Дискретная фильтрация

При частоте дискретизации – 204 Гц, длительности сигнала 1с, частоте сигнала 4 Гц, амплитуде и импульсной характеристики представленный на рисунке 34 будет сформирован сигнал, представленный на рисунке 35.

Рисунок 34 – Импульсная характеристика

Рисунок 35 – Дискретная фильтрация

ЗАКЛЮЧЕНИЕ

В результате выполнения работы по проектированию лабораторного онлайн-стенда были решены следующие задачи:

- изучены основы цифровой обработки сигнала;
- применены на практике основы программирования языка JavaScript и языка разметки и гипертекста HTML;
- изучена и применена библиотека chart.js;
- изучен и применен фреймворк vue.js;
- разработан сервис для генерации сигнала с заданными параметрами, с возможностью проведения частотного анализа.
- изучен фреймворк Ionic 3 и разработано мобильное приложение
- изучен фреймворк Express.js с помощью которого реализована серверная часть приложения.

Все математические операции и графическое представление реализованы с помощью JavaScript.

Онлайн-стенд является отличным сервисом для выполнения работ по дисциплине ЦОС. Данный сервис предоставляет методы генерации сигналов, с возможностью проведения спектрального анализа для оценки влияния на спектр различных параметров. Также разработано мобильное приложение, позволяющее использовать его функции для мобильных устройств.

Основными преимуществами онлайн-стенда являются:

- Кроссплатформенность
- Доступность в любое время

В дальнейшем система может быть дополнена новыми функциями, такими, как модуляция, синтез фильтров.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Цифровая обработка сигналов [Электронный ресурс] – режим доступа: http://files.lib.sfu-kras.ru/ebibl/umkd/50/u_lectures.pdf
2. Википедия – свободная интернет-энциклопедия [Электронный ресурс] – режим доступа: https://ru.wikipedia.org/wiki/
3. Основы HTML [Электронный ресурс] – режим доступа: http://html5book.ru/osnovy-html/
4. Основы CSS [Электронный ресурс] – режим доступа: http://html5book.ru/osnovy-css/
5. Основы JavaScript [Электронный ресурс] – режим доступа: http://html5book.ru/osnovy-javascript/
6. JpGraph – как это работает [Электронный ресурс] – режим доступа: http://jpgraph.ru/
7. Built with Bootstrap [Электронный ресурс] – режим доступа: http://getbootstrap.com/
8. Руководство Vue.js [Электронный ресурс] – режим доступа: https://ru.vuejs.org/v2/guide/index.html
9. Leverage Existing iOS Views In Your React Native App [Электронный ресурс] – режим доступа: http://moduscreate.com/leverage-existing-ios-views-react-native-app/
10. Ionic Documentation [Электронный ресурс] – режим доступа: https://ionicframework.com/docs/
11. Руководство Express.js [Электронный ресурс] – режим доступа: http://expressjs.com/ru/guide/routing.html
12. Vue.js Documentation [Электронный ресурс] – режим доступа: https://vuejs.org/v2/guide/

ПРИЛОЖЕНИЕ А

signal-generation.html

```
<ion-header>
  <ion-navbar>
    <button ion-button menuToggle>
      <ion-icon name="menu"></ion-icon>
    </button>
    <ion-title>Генерация сигнала</ion-title>
  </ion-navbar>
</ion-header>

<ion-content padding>
  <ion-grid>
    <ion-row>
      <ion-col col-12>
        <ion-list>
          <ion-item>
            <ion-label floating>Частота дискретизации</ion-label>
            <ion-input [(ngModel)]="ch_d" type="number"></ion-input>
          </ion-item>
```

```
<ion-item>
  <ion-label floating>Длительность сигнала (с.)</ion-label>
  <ion-input [(ngModel)]="time" type="number"></ion-input>
</ion-item>

<ion-item>
  <ion-label floating>Частота сигнала</ion-label>
  <ion-input [(ngModel)]="ch_s" type="number"></ion-input>
</ion-item>

<ion-item>
  <ion-label floating>Постоянная составляющая</ion-label>
  <ion-input [(ngModel)]="consta" type="number"></ion-input>
</ion-item>

<ion-item>
  <ion-label floating>Амплитуда</ion-label>
  <ion-input [(ngModel)]="a" type="number"></ion-input>
</ion-item>

<ion-item style="padding-top: 25px;">
  <ion-label>Вид графика</ion-label>
```

```
                <ion-select [(ngModel)]="menu" cancelText="Отмена">
                  <ion-option value="sin">sin</ion-option>
                  <ion-option value="square">square</ion-option>
                  <ion-option value="saw">saw</ion-option>
                </ion-select>
              </ion-item>
            </ion-list>
            <div class="content-button center">
              <button class="generate-button" ion-button
(click)="generateSignal()">Сгенерировать</button>
            </div>
          </ion-col>
        </ion-row>
        <ion-row>
          <ion-col col-12>
            <canvas #barCanvas></canvas>
          </ion-col>
        </ion-row>
      </ion-grid>
    </ion-content>
```

genaration-signal.ts

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import * as config from '../../config';

@Injectable()
export class GenarationSignalProvider {

  constructor(public http: HttpClient) {
    console.log('Hello GenarationSignalProvider Provider');
  }

  public getSignalParams(ch_d, time, ch_s, consta, a, menu){
    return this.http.get(`${config.config.devUrl}/generate-signal?ch_d=${ch_d}&time=${time}&ch_s=${ch_s}&consta=${consta}&a=${a}&menu=${menu}`);
  }

  public getWindowParams(ch_d, time, ch_s, consta, a, menu, window_function){
    return this.http.get(`${config.config.devUrl}/window?ch_d=${ch_d}&time=${time}&ch
```

```
_s=${ch_s}&consta=${consta}&a=${a}&menu=${menu}&window_function=${
window_function}`);
        }

        public getAnalysisSignal(ch_d, time, ch_s, consta, a, menu,
window_function, porog_amp){
          return this.http.get(`${config.config.devUrl}/signal-
analysis?ch_d=${ch_d}&time=${time}&ch_s=${ch_s}&consta=${consta}&a=${
a}&menu=${menu}&window_function=${window_function}&porog_amp=${por
og_amp}`);
        }
      }
```

signal-generation.ts

```
      import { Component, ViewChild } from '@angular/core';
      import { IonicPage, NavController, NavParams } from 'ionic-angular';
      import { GenarationSignalProvider } from '../../providers/genaration-
signal/genaration-signal';
      import { Storage } from '@ionic/storage';
      import { Chart } from 'chart.js';

      @IonicPage()
      @Component({
```

```
  selector: 'page-signal-generation',
  templateUrl: 'signal-generation.html',
})
export class SignalGenerationPage {
  @ViewChild('barCanvas') barCanvas;

  barChart: any;

  public response: any = {response: ''};

  public ch_d: any;
  public time: any;
  public ch_s: any;
  public consta: any;
  public a: any;
  public menu: any;
  public params: any;
  public ticks: any;

  constructor(public navCtrl: NavController, public navParams: NavParams, public genarationSignalProvider: GenarationSignalProvider, private storage: Storage) {
  }
```

```
ionViewDidLoad() {
  this.ch_d = 200;
  this.time = 1;
  this.ch_s = 4;
  this.consta = 0;
  this.a = 1;
  this.menu = "sin";
}

generateSignal() {
  this.genarationSignalProvider.getSignalParams(this.ch_d, this.time,
this.ch_s, this.consta, this.a, this.menu).subscribe((response) => {
    this.params = response;
    this.ticks = this.generateTicks();
    this.saveParams();
    this.barChart = new Chart(this.barCanvas.nativeElement, {
        type: 'line',
        data:{
          datasets: [{
            label: 'Генерация сигнала',
            data:  this.params,
```

```
backgroundColor: [
                  'rgba(255,255,255, 0)'
                ],
                borderColor: [
                  '#ff4c70'
                ],
                borderWidth: 1,
                pointRadius: 0
              }]
            },
            options: {
              scales: {
                xAxes: [{
                  type: 'linear',
                  position: 'bottom',
                  scaleLabel: {
                    display: true,
                    labelString: "Время, с",
                    fontColor: "black"
                  },
                  ticks: {
                    min: this.ticks.x.min,
```

```
                max: this.ticks.x.max,
                stepSize: this.ticks.x.step
              }
            }],
            yAxes: [{
              type: 'linear',
              position: 'left',
              scaleLabel: {
                display:  true,
                labelString: "Амплитуда",
                fontColor: "black"
              },
              ticks: {
                min: this.ticks.y.min,
                max: this.ticks.y.max,
                stepSize: this.ticks.y.step
              }
            }]
          }
        }
      });
});
```

```
}

saveParams(){
  this.storage.set('SignalParams', {
    ch_d: this.ch_d,
    time: this.time,
    ch_s: this.ch_s,
    consta: this.consta,
    a: this.a,
    menu: this.menu
  });
}

generateTicks() {
  return {
    x:{
      min: -0.2,
      max: Number(this.time) + 0.2,
      step: 0.2
    },
    y:{
      min: this.a - (2*this.a) - 0.5 + this.consta,
```

```
        max: Number(this.a) + 0.5 + this.consta,
        step: 0.5
      }
    };
  }
}
```

ПРИЛОЖЕНИЕ Б

server.js

```
var express = require('express');
var app = express();

const cors = require('cors');

app.use(cors({
    origin: function (origin, callback) {
        callback(null, true)
    },
    credentials: true,
        methods: ['GET'],
}));

app.get('/generate-signal', function(req, res){
  require('./generation-signal.js')(req, res);
});

app.get('/window', function(req, res){
  require('./window.js')(req, res);
});

app.get('/signal-analysis', function(req, res){
  require('./signal-analysis.js')(req, res);
});
```

```
app.listen(3000, function () {
  console.log('Example app listening on port 3000!');
});
```

signal-analysis.js

```
module.exports = function(req, res){

  var complex = require('./library/complex.js');
  var real = require('./library/real.js');

  var menu = req.query.menu;

  const ch = req.query.ch_d;
  const time = req.query.time;
  const chs = req.query.ch_s;
  const consta_s = req.query.consta;
  const a = req.query.a;
  const consta = Number(consta_s);
  const porog = req.query.porog_amp;
  const window_function = req.query.window_function;

  var const_window = 1;
  var A = Number(a);
  var f1 = new Array();
  var wind = new Array();
  var cosPoints_multi = new Array();
  var result = new Array();
  var t;
```

```
        if(menu == "sin"){
          for (t = 0; t < time; t += 1. / ch) {
            f1.push({x: t, y: consta + A * Math.sin(2 * Math.PI * t * chs)});
            if(window_function == "constanta"){
              wind.push({x:t, y:const_window});
            }else if(window_function == "heming"){
              wind.push({x: t, y: 0.54 - 0.46 * Math.cos(2 * Math.PI * t / time)});
            }else if(window_function == "natalla"){
              wind.push({x: t, y: 0.3635819 - 0.487396 * Math.cos(2 * Math.PI * t /
time) + 0.144232 * Math.cos(4 * Math.PI * t / time) + 0.012604 * Math.cos(6 *
Math.PI * t / time)});
            }
          }
          for(var key in f1){
            cosPoints_multi.push([f1[key].x, wind[key].y * f1[key].y]);
          }
          for(var key in f1){
            result.push(wind[key].y * f1[key].y);
          }
          var fft = new complex.FFT.complex(result.length, false);
          var r1 = new Array();

          fft.simple(r1, result, 'real');
          var cosPoints = [];
          for (var i = r1.length / 2; i < r1.length; i += 1) {
              cosPoints.push([i - r1.length - 2, Math.abs(r1[i])]);
          }
          for (var i = 0; i < r1.length / 2; i += 1) {
              cosPoints.push([i, Math.abs(r1[i])]);
          }
```

```
var max = 0;
for (i = 0; i < cosPoints.length; i++) {
   if (cosPoints[i][1] > max){
     max = cosPoints[i][1];
   }
}

for (i = 0; i < cosPoints.length; i += 1) {
   if (cosPoints[i + 2][1] < porog * max) {
      cosPoints[i].shift();
   } else {
      break;
   }
}

for (i = cosPoints.length; i > 0; i -= 1) {
   if (cosPoints[i - 2][1]  < porog * max) {
      delete cosPoints[i]
   } else {
      break;
   }
}
var otschet = 0;
max = 0;
for (var key_new in cosPoints) {
   if (cosPoints[key_new][1] > max) {
      max = cosPoints[key_new][1];
      otschet = cosPoints[key_new][0];
   }
```

```
    }

    var vrem = otschet / chs;
    for (var key_new1 in cosPoints) {
        cosPoints[key_new1][0] = (cosPoints[key_new1][0]) / vrem;
    }

    var result = [];
    for (var k in cosPoints){
      result.push({x: cosPoints[k][0], y: cosPoints[k][1]});
    }
  }
  res.send({fft: result});
}
```